AN EMPIRICALLY DEVELOPED COMPETITION-BASED METHOD FOR SOFTWARE REQUIREMENTS PRIORITIZATION PRACTICE: A GROUNDED THEORY APPROACH

Khalid S. ALWAHABI

SOFTWARE ENGINEERING

JUNE 2017

AN EMPIRICALLY DEVELOPED COMPETITION-BASED METHOD FOR SOFTWARE REQUIREMENTS PRIORITIZATION PRACTICE: A GROUNDED THEORY APPROACH

BY

Khalid S. ALWAHABI

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

SOFTWARE ENGINEERING

JUNE 2017

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN- 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **Khalid S. ALWAHABI** under the direction of his thesis advisor and approved by his thesis committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN SOFTWARE ENGINEERING**

Dr. Mahmood K. Niazi
(Advisor)

Dr. Khalid A. Aljasser
Department Chairman

Dr. Sajjad Mahmood
(Co-Advisor)

Dr. Salam A. Zummo
Dean of Graduate Studies

Dr. Jameleddin. Hassine
(Member)

8/6/17

Date

DEDICATION

I dedicate this effort to my parents, family for all kind of support. I would like to pass a message to my sons that knowledge is a power; education is an opportunity that will not be given but must be taken. a person should invest on it throughout his/her entire career life.

ACKNOWLEDGMENTS

Firstly, I would like to express my appreciation to my thesis advisor Dr. Mahmood Khan Niazi for the continuous support and guidance during my thesis work.

Second, I would like to thank my thesis committee: Dr. Sajjad Mahmood and Dr. Jamaluddin Hassine, for their, valuable comments and feedback.

I would like also to thank my parents for all kinds of encouragement and pray for me at the first place and supporting me throughout my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

SE		Software Engineering
RP	:	Requirements prioritization
RE		Requirements Engineering
DM	:	Decision-making
SLR		systematic literature review
E		Elements (requirements)
GT		Grounded Theory
gP		grid property
SDLC		Software Development Life Cycle
E		Requirement elements (featureS)
EgP		Requirement elements values for properties grid.
CG		Number of competition grids
gpn:		Number of properties

ABSTRACT

Full Name : Khalid S. ALWAHABI

Thesis Title : AN EMPIRICALLY DEVELOPED COMPETITION-BASED METHOD FOR SOFTWARE REQUIREMENTS PRIORITIZATION PRACTICE: A GROUNDED THEORY APPROACH

Major Field : Software Engineering

Date of Degree : JUNE 2017

Requirements prioritization (RP) is a critical practice of the software engineering process, in which customer's requirements are developed in sequence for software release planning purposes. Typically, requirements prioritization is carried out from one of three aspects, namely, technical, business or client, without quantifying them holistically. This often results in inconsistencies between the software deliverables and product release planning.

A new competition-based method for requirements periodization practice has been developed to address this issue during the requirement engineering process phase for commercial system development in the Oil & Gas industry. An observation study was conducted using the grounded theory as a research method in observing and exploring aspects of the practice. The proposed framework showed a significant impact in controlling the consistency issue and promises to benefit companies and organizations involved with software development.

In this research work, highly considered requirements prioritization concerns were identified in the following order; 1) Review the literature and de-facto software development industrial practices, 2) Investigate inter-relationships between the identified

properties to classify pairs of properties that can contribute to the competition-based matrix, 3) Conduct an empirical study to observe the requirements prioritization practices for three software development companies using the grounded theory approach, 4) Evaluate the quality attributes of the proposed framework among other requirements prioritization techniques including (*concept, ease of use, and size of data, fuzziness, multi-criteria, stakeholders-involvement, and complexity*),

ملخص الرسالة

الاسم الكامل: خالد سعد علي الوهابي

عنوان الرسالة: ترتيب أولويات تنفيذ متطلبات البرامج الحاسوبية باستخدام قاعدة تحديد المقارنات وتبيين المفارقات.

التخصص: هندسة البرمجيات

تاريخ الدرجة العلمية: رمضان 1438

تعد عملية تحديد أولويات تنفيذ متطلبات البرامج الحاسوبية ممارسة مهمة في عملية هندسة المتطلبات البرمجية التي يتم فيها تطوير متطلبات العميل بالتتابع لغرض تحقيق تخطيط ملائم لخطة تطوير المشروع البرمجي.

عادةُ، يتم تحديد أولويات المتطلبات من جانب واحد من ثلاثة جوانب مختلفة، أي الجانب التقني والجانب التجاري وجانب العميل دون الأخّذ في الاعتبار تحديدها بشكل شمولي من جميع الجوانب، مما يؤدي إلى مشكلة عدم التوافق بين ما تم تطويره فعلياً وماهي عليه عمل خطة التطوير.

وبالتالي قمنا في هذا البحث بدراسة جوانب المشكلة واقتراح مفهوم جديد في هندسة البرمجيات "تمييز البرمجيات" باستخدام قاعدة تحديد المقارنات وتبيين المفارقات والتي تتم عبر تنفيذ اختبارات تنافسية قائمة على مزايا مختارة لتمييز بشكل جذري للبرمجيات والذي يعتبر أجراء مهم خلال مرحلة عملية الهندسة البرمجيات.

في هذا البحث تمت الدراسة والتجربة لهذا المفهوم في تطوير نظام معلوماتي معقد يستخدم في صناعة النفط والغاز. وقد أجرينا دراسة بحثية تجريبية في الموقع باستخدام البحوث النظرية المستندة إلى الرصد والتدوين والمعروف ب (grounded theory) والذي مكننا من استكشاف مكامن التحديات والخلل التي يجب معالجتها. والتحقق بملائمة الاختبارات التنافسية والعلاقة المتبادلة بين الخصائص المراد تميز البرمجيات بناء عليها والتي تسهم في الأخير في تسهيل عملية تحديد الأولويات ا عند جدولة خطط تنفيذ المتطلبات البرمجية وطرحها للمستخدم النهائي.

أظهرت نتائج البحث مثالية هكذا نموذج المقترح قابل للاستخدام بسهولة والأثر الكبير في التحكم في مسألة التوافقية بين ما يتم التخطيط لتطويره وما تم تطويره فعلياً. يعد هذا المفهوم جديدُ في مجال هندسة البرمجيات واعد في خدمة شركات التطوير البرمجي والعمل البحثي.

CHAPTER 1: INTRODUCTION

Background

The complexity of software systems has increased exponentially and the implementation of new requirements has taken a form of incremental development process. Requirements prioritization practice (RP) is performed by selecting and ranking the requirements according to their importance and planning their implementation in successive releases based on their priority. In recent research studies [9, 11, 16], requirements prioritization practice has become a complex decision-making process that has gained a lot of interest with researchers seeking to satisfy the demands of stakeholders.

Prioritizing requirements is a key practice of requirements selection during the decision-making process that focuses on the deliverables, which ultimately adds value to the product when considering time, cost, and other constraints. The prioritization process can sometimes lead to conflict between project stakeholders in terms of their preferences of requested features.

It is therefore necessary, before we find the phase of performing requirements prioritization practice, we need to first understand the requirements engineering (RE) process to see where it fits and how RP practice can be utilized during the release planning process.

The main part in the RE process is the software requirements (features) which is a description of software functionalities. The requirements can be obvious or hidden, known or unknown, expected, or unexpected from the client's point of view [2]. Thus, the process that is necessary to gather the software requirements from the client, analyze and document these requirements is known as the requirements engineering process.

The RE process takes place at the beginning of every software development project, and produces a result in the form of specifications that define the product to be developed. This approach is based on the Waterfall model [50] where requirement engineering is followed by design, implementation, and testing maintenance activities. However, this cascading process may not be the most proper in practice, since the flexible nature of software requires the development process to be more iterative and evolutionary. New or changed requirements appearing during development calls for continuous RE efforts. The goal of requirements engineering is to develop the descriptive 'System Requirements Specification' document. The requirements must be found and agreed to by customers, users, and suppliers before the software can be built.

The four main steps in the process, as defined by Sommerville [2], for requirements engineering includes:

- Feasibility study: the goal of this step is to analyze whether the software product can be implemented, the project's contribution to the organization, and cost

constraints. It also explores technical aspects of the project and product such as usability, maintainability, productivity, and integration ability. The outcome of this step is a report that holds enough information and recommendations for management to determine whether the project should be carried out or not.

- Requirements gathering: is the step to collect user requirements and communicate with the client or end-users to understand what the software should offer and which features they want the software to include.
- Software requirements specification: the systems analyst creates a document from the requirements collected from the various stakeholders. This document defines how the intended software will interact with hardware, external interfaces, speed of operation, response time of system, portability of software across various platforms, maintainability, speed of recovery after crashing, security, quality, limitations etc.
- Software requirements validation: is the last step of the RE process which validates the analyzed requirements in terms of its functionality, domain of software, ambiguities, and completeness.

With the completion of all the above RE process steps, the technical implementation starts with a plan, usually called development plan. In fact, this plan should be aligned with the actual software release plan or commercialization plan.

Software release planning is a process of selecting a set of software requirements (features) to deliver in a release within a given period. The purpose of release planning is to balance between competing stakeholders' demands and benefits under technical and non-technical constraints. It is an implementation of consequence updates and feedback from customers,

defects in the previous release, market factors, new customer demands and other technical and non-technical requirements.

Requirements prioritization practice is considered as the main complementary part of the release planning process. It concerns about the assignment of requirements in a sequence of releases to help improve a software product in terms of different attributes by adding new features or improving some quality aspects within predefined technological constraints [8]

1.1 Research Problem Overview

Several requirements prioritization approaches have been proposed that are designed to work on different measurement scales, attributes, levels of complexity, and vary in their abstraction levels form high to low level of details [24]. Most proposed RP practices have been evaluated in terms of their ease of use and performance, as well as toleration of faults [24]. However, the basic issue that has not gained a solid resolution is the inconsistency between the software deliverables and product release planning, which defines the reliability attribute that software deliverables should be aligned with the release plan as well as other attributes such as comprehensiveness, simplicity, practicality and the ease of use.

Figure 1-2: shows the various components in the software release planning framework that help us to figure out the relative percentage influence of each component. Release planning is a framework of several components that directs or streamlines the RP process.

There are three main attributes: release objective, resources allocation and requirement prioritization that control the release planning process. Release objective is considered to

plan deliverables based on targeted objectives, for example fix software bugs that appear in the latest software release or that will benefit the market volume, where if it is based on the available resources, skill set or budget, will need to be allocated. On other hand considering the requirements prioritization will have the focus on the characteristics of the feature's added value, complexity, stakeholders, and cost.

Figure 1-1: Size of the release planning components

Software development vendors try to improve their software products by understanding the new requirements and then implementing them in a well-defined order for future software releases. Prioritizing and supporting a large set of requirements is a challenging decision-making process where many factors need to define such as the requirement purpose, development complexity and technical feasibility, risk, cost and added value and so on.

There is a growing acknowledgment in industrial software development that requirements are of varying importance. Yet there has been little progress to date, either theoretical or practical, on the mechanisms for prioritizing software requirements [2].

Building a comprehensive, practical, easy to use RP method is the research goal that discussed how to apply the best research method that can guide us to identify accurately the concepts and components of the problem solution.

Therefore, the concern is how to develop an efficient requirements prioritization practice that produces a list of prioritized requirements aligned with the release planning process. This can be achieved when a comprehensive approach is performed to explore the characteristics of each requirements feature, which is the missing key aspect in all the reviewed literature and surveyed industrial RP practices.

1.2 Research Motivation

The issue of inconsistency between software deliverables and product release planning is considered as one of the most substantial problems in software development. A review of some studies shows that there is no widespread practice for RP [28]. Also, there is no standard practice that has gained enough acceptances among software development companies. In fact, the inconsistency issue in release planning explored in the research studies stated that issue is the only outstanding challenge of release planning that has not been thoroughly addressed. Thus, building a standard process to address that challenge involves a lot of process to be considered in organizational practices, processes, and people. the problem also has been acknowledged in the literature as one of the major causes of software project delay/failure.

Therefore, the problem is not with the release planning itself, but with limited aspects during the prioritization practice for handling it. In fact, it is well acknowledged in literature that there is an increasing need for a well-established a multi-criteria decision making process for prioritizing the requirements [5] [10] [13] as well as a methodological approach for quantifying evolvable software requirements [17]. Establishing a comprehensive RP practice for handling several aspects help supporting the software requirements development plan consistency and increases the likelihood of project success.

1.3 Research Method

In this research work, we follow the design science paradigm [34] that defines an applied research workflow as guidelines.

We illustrate how authors, reviewers, and editors can apply those guidelines as a model to formulate a way of conducting, evaluating, and presenting the new developed software improvement process.

The following are the research design guidelines presented in [34]:

Guideline 1: Design as an Artifact: the research design must be conducted in the form of a model or method.

Guideline 2: Outstanding issue: the research should have an objective to address important problems.

Guideline 3: Design evaluation: show applicability through a case study or real experiment.

Guideline 4: Research rigor: show the research model applies the right methods in construction and evaluation of the design artifact.

Guideline 5: Design as a search process: well-defined search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem domain [34].

Guideline 6: Communication of research: the research should be presented for technology/management-oriented audiences

Therefore, the research work will be aligned to the following guidelines:

1- Design as an artifact: the main artifact of our research work is the competition-based characterization method that guarantees an optimal and practical software requirement prioritization practice. This competition-based characterization method is typically applied in the medical treatment field where medical personnel prioritize patients based on disease criticality vs treatment. This method helps quantify requirements using several measurement criteria found for a software development plan and serves as a core component within release planning framework. A grounded theory approach [16] will be used as a research method.
2- Outstanding issue: Our research problem is highly relevant to the field of software requirements engineering and specifically release planning. Requirement prioritization practice is critical and serves as the main component towards the success of the release planning process [6, 8, 10, and 24]. Furthermore, requirements prioritization should be calculated iteratively in a comprehensive way, where several measurements are considered when deciding

selection/prioritization. It has been highlighted as challenge [24] to solve two conflicting prioritization criteria. In this paper, we focus on addressing this issue as a part within the proposed framework in the context of a real software industry issue.

3- Design evaluation: The proposed framework is evaluated in two different scenarios using a real dataset to prove the concept, and a real dataset to simulate a real release planning scenario to confirm the validity

4- Research rigor: In this research work, we present a competition-based characterization method that uses a strategic quadrants grid as a model of classification and scoring for each software requirement feature. The workflow of which is tested against two interrelated release planning objectives, where each grid cell is ranked according to the importance of each intersection. Then, accumulated ranks are calculated for each software feature before sorting them to find the highest number with the highest priority. Another part of the research evaluation model is performed to test the consistency, scalability, error proneness, computational complexity, reliability, and comprehensiveness for the subjective prioritized requirements verses our proposed approach result.

5- Design as a search process: The search process of the design presented in this research work is an iterative search for recent research advancements in the field of software requirements prioritization. Two main components were developed within this proposed approach for requirements classification and prioritization. This provided the release planning process with clear-cut requirement artifacts which

were empirically evaluated for optimal construction purposes and internal/external impact and applicability purposes.

6- Communication of research: The main audience of this research work will be software project managers who oversee a release planning process.

7- Communication of research: The main audience of this research work will be for software project manager who oversee release planning process.

1.4 Research Contributions

This research work has the following contributions:

1. Conducted an empirical study to observe the requirements prioritization practice using the grounded theory as a method to study and report the findings of requirements prioritization concerns and issues.

2. Proposed a Requirement characterization framework for requirement prioritization practice which uses a competition-based technique applied in the medical field to prioritize patients for urgency of treatment.

3. Investigated and analyzed the impact of the identified RP parameters towards the success of release planning.

4. Explored the relationship factors between interrelated RP parameters.

5. Evaluated the applicability of the proposed framework using small and real datasets.

6. Developed a software tool to automate the workflow of this proposed model and calculate the prioritized requirements list.

7. Conducted a comparative analysis of the proposed framework with other commonly practiced RP methods using the following comparison criteria;

- Concept: Indicates how a method applies the processes, following systematic and analytical approaches, rules, and recommendations employed by the prioritization method.
- Ease of Use: Indicates a characteristic of how a new user can easily use and apply the prioritization method.
- Size of data (Scalability): Measures the amount of requirement sets accommodated by the prioritization method to minimize human effort.
- Fuzziness: Measure the uncertainty of human thought related with the implementation of the prioritization method.
- Multi-criteria: Refers to the multiple factors considered during the prioritization method practice.
- Stakeholders-involvement: Refers to the multiple stakeholders involved with the application.
- Complexity: Measures the number of comparisons required to execute the prioritization method.

8. Prepared three papers titled as below for journal/conference publication.

- Competition-Based Requirements Characterization Method for A Comprehensive Software Requirement Prioritization Practice;
  Khalid S. ALWAHABI, Dr. Mahmood K. Niazi
- Requirements Prioritization Overview in Market-driven Software Development – A Grounded Theory Study
  Khalid S. ALWAHABI, Dr. Mahmood K. Niazi
- Evaluating the practical use of requirements characterization framework for requirements prioritization practice: A CASE STUDY;
  Khalid S. ALWAHABI, Dr. Mahmood K. Niazi

1.5 Thesis Structure

The thesis documentation is organized as follows:

- Chapter 1: introduces the research topic. It covers the background, challenges, description of the research problem and study design method, and a list the research contributions.
- Chapter 2: explores the literature reviewed regarding requirements prioritization methods, release planning models, applied measurement factors and some other related work.
- Chapter 3: documents the research methodology process and illustrates the applied research methodology in detail. Subsequently, this chapter presents the study results and provides a discussion on implications of the findings of the theory and the practice.
- Chapter 4: presents the proposed RP model and applies it using real data.
- Chapter 5: presents the comparative analysis as an adopted evaluation framework, discussed the result
- Chapter 6; concludes the research work findings with a summary, contributions, limitations and future work.
- Appendix include the following:
    - A description of *CharFramework* software tool for the proposed framework.
    - Miscellaneous related references.
    - Experiment dataset

CHAPTER 2: LITERATURE REVIEW

The purpose of this chapter is to provide an overview of existing research conducted in the context of requirement prioritization for software release planning. This literature review provides the reader with an idea of how previous research addresses this problem, how different research approaches can be related to each other and how they influenced this thesis. It also provides the readers with information about problems and limitations of the previous research work. In the next subsection, we focused on the four main topics of importance to our research problem during the review process:

- Requirement prioritization practice
- Prioritization aspects
- Requirement prioritization techniques and the validation process
- Release planning models
- Requirements Prioritization Parameters

2.1 Requirements Prioritization Practice

Requirements prioritization plays a crucial role in software development, and it allows for planning software releases, combining strategies for budget management and scheduling,

as well as market strategies. It is, in fact, considered a complex multi-criteria decision-making process [7].

RP practice is performed during requirements analysis process [6]. It is an area of optimization for software development process involving many stakeholders to maximize the utilization of limited resources and map them into business benefits [4]. It is an essential part of release planning process that has some direct/indirect impacts on software marketing process [11]. Thus, the importance of this practice gives us an advantage to plan on ahead of time and address all related concerns considering limited resources, inadequate budget, and insufficient skilled developers.

Requirements prioritization practice is an iterative process [5] and might be performed at different abstraction levels and with different information in different phases during the software lifecycle. Many theoretical and practical requirements prioritization techniques were proposed in literature to handle different challenges related to support the decision of prioritizing software requirement features [1, 2, 4, 16, 19, 22].

Philip et al. [34] conducted a systematic literature review by exploring all RP techniques and release planning methods, highlighted the objectives, challenges, and limitations of each technique/method. There was no any effort to tackle the inconsistency issue in al techniques. Joans defined et al. [7] challenges as "*A study of identifying and choosing alternatives based on the values and preferences of the decision maker*". Several challenges associated with planning, designing, building, deploying, and testing releases such as ensuring each release is managed effectively and delivered efficiently. It is usually

impossible to implement all the requirements due to limited resources in terms of budget, staff, and schedule.

Amir et al. [9], also found some potential challenges that are related to the release planning process optimization, including people *cooperation, disciplines, abilities, systematic approaches, resources constrains, complexity, and interdependency requirements*. Moreover, a study [9] confirmed that there is a strong correlation between the requirements prioritization practices and the rate of software product release delays [31].

Lubars et al. [10] found that many organizations believe that it is important to assign priorities to requirements and to make decisions about them according to rational, quantitative data [3]. Still it appeared that no company really knew how to assign priorities or how to control the consistency of the requirements development plan and commercialization plan.

The most critical practice in RE process is the selection aspects of the 'right' requirements and plans the releases; Research studies followed a certain workflow when introducing new methods/ techniques. The first step is concerned about the selection of the most appropriate prioritization criterion. Second step is the identification of requirement attributes that define the ranking process.

Therefore; it is a mandatory action for project managers to select the appropriate method that quantify the requirement features and that uses criteria to prioritize those requirements, as well as the need of a domain knowledge expertise [22] to perform the RP practice.

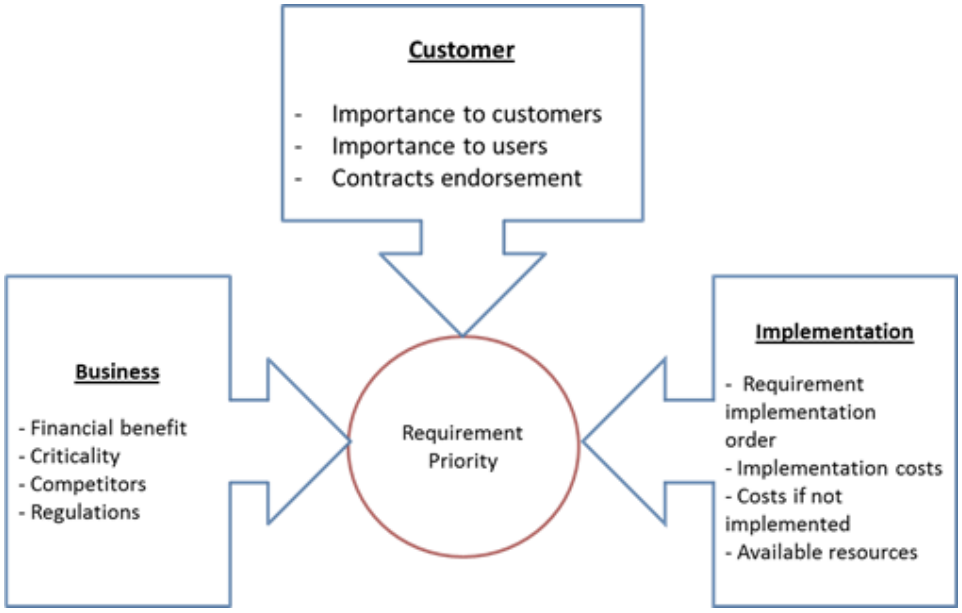


Figure 2-1: Requirement priority aspects

2.2 Prioritization Aspects

The success of quality software development depends on the right selection of candidate requirements which are prioritized based on key priority aspects [10]. Prioritization aspect is defined as set of properties that defines a measurement value of requirement *E*. There are many concerns in defining which aspects [33] should be considered during prioritization practice. Figure 2-1: shows the three aspects associated with their properties consolidated from the literature [4][6][10].

Typically, requirements prioritization is carried out from three different aspects, namely, technical, business and client. Many techniques facilitate the requirements prioritization process through technical aspects. However, VOP is the only technique which covers five business aspects and one technical aspect for requirements prioritization process Most of RP techniques are usually evaluated based on business or technical aspects [35], Table 2-1: shows different aspects with explored properties of these RP techniques.

Technical aspects	Business aspects	Client aspects
Consistency Ease of use Time complexity Accuracy Error proneness Scalability Reliability Comprehensiveness Software bug Reuse frequency Technical feasibility Volatility	Benefit/value Cost Strategy alignment Risk Human Resources Competitive Innovative	Customer Satisfaction Importance Urgency

Table 2-1: RP aspects properties

Amir et al [9] listed the following steps as guidelines for software requirements decision making process:

1- Identify the challenges and its related boundaries
2- Establish goals and system objectives
3- Put conditions/requirements that problem solutions must meet
4- Setup alternatives
5- Define criteria to measure the goals and differentiate the alternatives.
6- Select the decision-making tool: the selection of a suitable tool is based on the objective of the decision-making approach (such as multi-voting or brainstorming).
7- Evaluate alternatives against criteria.
8- Validate solutions against the objectives of the problem.

Although, it is easy to consider one aspect as the focus of decision-making process by recognizing the requirements importance aspects to the system but it also can show an indication of RP process generalization problem [6]. Therefore, to define the requirements

priority from different aspects, it is highly complex decision-making activity but more reliable to where some the aspects' properties are interrelated and consider thoroughly during DM process. There are some business advantages in maximizing the profit, enabling the business growth, and gaining customer's attraction and satisfaction [15] of applying multi-aspects technique. The responsibility of requirements engineers is to select the proper methods, techniques, and tools to streamline the requirements prioritization process based with well-defined technical and business aspects [22].

At present, RP practice in software development industry [18] is described clearly in concept but not practically implemented. Stakeholders need to find a way to compare/quantify the requirements before they can be properly ranked or prioritized on a particular scoring system [19].As far as the selection of the RP techniques is concern, various prioritization techniques have been evaluated on the number of characteristics: concept, ease of use, Fault tolerance, multi-criteria, multi-person, speed and complexity analysis in [23] using comparative analysis.

2.3 Requirement Prioritization Methods and Techniques

Several methods and techniques have been proposed for software release planning process. Some are appointed towards the release planning problem while others are just requirements prioritization techniques. The process of comparing requirements to each other becomes complex as the number of requirements increase which introduce another challenge to the subject.

Karlsson et al. [4] suggests that RP process should be done in an incremental way as patches at the time of analyzing requirements set in which the analysts and stakeholders

find it easy to interact with each other and agree on a set of requirements which need to be implemented.

More research works [30 , 31] is conducted as a systematic review and mapping studies addressing some research questions in exploring and requirements selection factors, validation process and their impact towards release planning .Research findings , were complemented each other where researchers found several facts in terms of the main focus on the proposed methods that were limited to set of requirements selection factors, with an emphasis on constraints like cost, resource, effort, and time-related ; some of the methods were validated in the industry, rest was validated in the academia with 80% case studies and most of them were market-driven software development; and no proposed release planning used outside the scope of academic work [52]. Also, several systematic literature reviews [34] have been conducted to evaluate the effectiveness of some existing prioritization techniques and explore their limitations, aspects. Most SLRs highlighted that lack of an empirical evaluation for several requirements prioritization methods [30], [31].

Mauricio et al. [30] identified some research of interests such as the adaptation of multi-objective based RP technique, use of hybrid technique that can complement RP problem solving, requirements interdependency that needs to analyze the effects when the requirements rate of changes is high, and the usage of experiment with real data in large scale which can reflect some thoughts of an optimized RP process [37].

Certain constraints in resources, time, and limited budget are not possible to be considered completed for set of requirements development and delivered in a single release. It is important for a software system development acceptance to understand the requirements

and selecting the best suit of requirements within limited resources and well-defined deadlines [28].

There are various techniques available to support prioritization process and their efficiency depends on many parameters. There is a growing demand in the industrial software development that requirements management and planning is the key process towards the success of a software project which is a challenging task for product manager

Karlsson et al. [16] described the requirements prioritization practices in three steps:

- Preparation: Structuring of the requirements according to the principles of the used RP method.
- Execution: establishing the measurement criteria that help decision makers to do the prioritization with a consideration of all the measurement information.
- Presentation: share the results for evaluation purpose and modification

Some prioritization techniques are based on a quantitative assignment to different aspects of requirements [18], Brackett et al proposed a numeral assignment technique, NAT which is based on the principle that each requirement and each requirement is classified as mandatory, desirable, or inessential and assign "a number on a scale ranging from 1 to 5, where the numbers indicate: 5: Mandatory (the customer cannot do without it). 4. Very important (the customer does not want to be without it). 3. Important (the customer would appreciate it). 2. Not important (the customer would accept its absence). 1. Does not matter". This technique is commonly practiced due to the advantages of easy to be used, less time to rank, fewer steps to make decision and since it has close human interaction more accuracy is achieved, this technique however lacks of the ability to deal with large

set of requirements [28] as well as can't be adapted easily within the company strategies in terms of exploring some business or client aspects. Another approach introduced by Saaty [16] called Analytical Hierarchy Process. AHP is pair-wise comparing the classified set of requirements to determine which has higher priority and to what extent on is a ratio scale based RP technique that involves building a (ranking) of decision properties (requirements) system and then before making comparisons based on their importance and priority. As a fact, AHP is not suitable for large requirements set where the total number of comparisons to be performed in AHP are n (n-1)/2 pair-wise comparisons, where n is the number of requirements. Similar RP technique to AHP that works in hierarchal approach called Hierarchy-AHP [16] where requirements are structured in a hierarchy and it is considered as an enhanced version of AHP that addressed the drawback in AHP [53] of preventing it from scaling up on the number of requirements. It works with large or even medium number of requirements. It reduces number of comparisons as the requirements are not compared pair-wise. The only trade-off is that consistency check is reduced because the number of redundant comparisons is decreased which increase the chances of judgmental errors. Perini and Fondazione proposed a technique [53] which combines project's stakeholders' preferences with requirements ordering computed through machine learning techniques that claimed to reduce the stakeholders' preferences effort and keep high level accuracy of result estimates. however, this approach is not easy for automation be in a sense of the practicality.

Priority group's technique was introduced by Karlsson et al [3] as the name of the technique goes, its outcome is not the groups of requirements. It functions same as the numerical assignment technique but with a minor difference. Each requirement is assigned to a group

of high, medium and low. Within these groups, more three subgroups are created and process is repeated until a single requirement is left in each subgroup. In this way, a group of high priority requirements can be implemented first and a group of low priority ones can be postponed for later releases. The result of this technique can be represented using ordinal scale.

CBRank technique [24] which is kind of a mixed process that combines a step of a pairwise case selection and a step of a pairwise preference and has the capabilities to solve the scalability issue in with good accuracy of the final rank and elicitation effort. Similarly, Karlsson et al. [3] introduced minimal spanning tree, it does not contain redundant comparisons which reduces number of comparisons to n-1 than which in AHP is n*(n-1)/2. With n-1 comparisons a minimal spanning tree can be constructed. It gives relative ranking intensity of requirements but due to lack of redundancy judgement errors cannot be identified. The result of this technique can be represented on ratio scale as in the case of AHP. Another similar approach introduced by Karlsson called Bubble sort [3] is one of the simplest and basic techniques and It works same as AHP [16] technique it follows the activity of pairwise comparison concept where it is use to be used but its fundamental issues are the time complexity and scalability.

Binary search tree introduced by Karlsson et al [4] nodes are labeled as elements (requirements). The requirements are prioritized using BST algorithm. The root holds a requirement that will be evaluated. Then each requirement is compared to this root node requirement and if it is of low priority it goes to the left node and if it is of high priority it goes to the right node. This process continues until all requirements are inserted in their

proper places. It takes n log n comparisons with n number of requirements [16]. This technique can determine to what extent a requirement is important.

Based on the importance, numerical assignment for sorted requirements are placed in BS-tree nodes for ranking requirements [2].where requirements placed in the left subtree of a node are of lower priority than the node priority, and all requirements placed in the right subtree of a node are of higher priority than that node priority before prioritization is performed by comparing a selected node to in the top and select an unsorted requirement to be compared with it and carrying out repeatedly the process until no further node needs to be compared and at that time the requirement can be inserted into the right position. This approach in general is considered one of the best due to the capabilities to handle huge requirements set and easy to use with decent quality in results in addition to the satisfaction of some other business aspects like company strategy and added market value but its time complexity is very high O (n log n).

Cumulative voting [6] or 100$ method is to distribute 100 points over the requirements; it is easy and fast to be carried out but limited comprehensibility with high number of requirements and flat requirements hierarchy. The issue in this technique is that stakeholders might be biased by assigning all their points to only of the requirement to make that requirement most important. However, it is considered faster and outperforming AHP. Beck K. et al developed RP method call planning game [18] which prioritize requirements based a combination of numerical assignment technique and ranking technique together where stakeholders write the requirements on story cards and the cards are sorted in different groups. The different groups should have different names, “those without which the system will not function,” “those that are less essential but provide

significant business value," and "those that would be nice to have." This technique is easy to be practiced using large requirements sets with few steps to make decision but it looks there are no chances for software vendor to participate in this prioritization. In cost-value approach [4] is introduced to prioritize requirements based on their added value and implementation cost but the issue here is the scalability and the time it takes. Another prioritization technique that produces the result of prioritization based on integrated technical aspects with business aspects is Kano model [31] which considers a comparison of customer satisfaction versus technical excellence. Other techniques are based on subjective measures which drive the giving priorities to requirements by reaching agreement between stakeholders.

Berander et al identified commonalities and differences with regards to their characteristics for some prioritization practices [14]. Number of prioritization methods are useful in small-scale data set and could applicable to some specific circumstances. AGORA is an extension to Goal-Oriented Requirements Analysis Method [20], which uses a goal graph contribution values and preference matrices are determined and these preferences are represented in the form of a matrix and stakeholders attach the value subjectively which no doubt will have accurate results but with high difficulty to use and to considered some business objectives. Win-Win [11] or Theory-W supports the negotiation process to solve disagreements about requirements and is commonly applied due to the lack of business aspects support. Davis developed a multistep process, "Requirement Triage" [21] which estimating resources necessary to satisfy each requirement. Wiegers' Method [22] calculate the priority from the value of a requirement, costs and technical risks associated with its implementation. This approach shows a reliable performance in satisfying most of

technical aspects but lacks the granularity necessary to determine whether or not the requirement considers key business values. Table: 2-2 shows different RP techniques available in the literature and the size of the requirements they support.

RP technique	*Scale Type*	*Size of Requirements*
Numerical assignment	Ordinal	Medium, Large
Ranking	Ordinal	Medium, Large
Priority groups	Ordinal	-
Game planning	Ordinal	Medium, Large
Requirements triage	Nominal	Small
TopTen	Nominal	Small
QFD	Ordinal	Small, Medium
Cumulative voting	Ordinal	Small, Medium
VOP	Ratio	Medium, Large
TOPSIS	Ratio	Medium, Large
AHP	Ratio	Small
CaseBase	Ratio	Small
EVOLVE	Ratio	Medium, Large
IGA	Ratio	Medium, Large
RUPA	Interval	Medium, Large
Minimal spanning tree	Ratio	Medium, Large
HCV	Ratio	Large
Hierarchy AHP	Ordinal	Medium, Large
Bubble sort	Ordinal	Small

Table 2-2: RP techniques and size of requirements

Therefore, prioritization techniques should be easy to use" [14], [18],[35], should put a confidence of the user on the system. Another fuzzy prioritization framework used practically [23] to elicit stakeholders' business goals, rate the stakeholders, allow the stakeholders to rate the importance of the requirements, as well as rate the requirements based on objective measure. Bagnall et al [27] proposed NRP methodology that allows a set of requirements to be released in an incremental way based on a company budget and meeting the demands more advancement on this this selection technique has been introduced to consider multi-factors [26] when making Multi-objective release planning.

MONRP can merge multiple, conflicting objectives to maximize customers' satisfaction and minimize the total effort involved in the development of the selected requirements. Therefore, achieving high-quality software is conditioned by the consideration all possible aspects during the RP practice.

2.4 Release Planning Models

The most crucial decision is to select a feature for implementation in the next software release. Many software release planning models are available which considers a wide variety of factors in deciding the implementation of a feature in a release [48]. All models provide different solutions of release planning and discuss different requirements selection factors. There are more than twenty-five release planning models proposed and systematically reviewed [7].

Manju et al. [25] conducted an industrial survey and highlighted the current market practice of release planning is only considering one or two properties for requirements prioritization process. In recent research advancements in software requirement selection, more attributes have been considered to support the decision making process by implementing and automating search-based methods serve as that utilize search-based optimization algorithms an alternative way to solve some software requirements selection and make more advantages towards the enhancement of the scalability, practicality, generality, robustness [26] for or a very large and complex requirement data set.

In section below, we highlight some information about RP models and their used selection factors.

Cost-to-Value Approach (CVA)

This model [48] focus on prioritizing software requirements based on stakeholders reference. It prioritizes the requirements based on their subjective relative judgment. It very simple put not used.

EVOLVE-FAMILY

This model considers the distribution of requirements to releases, controls stakeholder conflict, and allocate the resources to all the releases. It is an iterative approach which offers decision support for release planning. It is well known and consider the root version of EVOLVE-family [46]. Another version of this EVOLVE-family is called Evolve+ which an extension of EVOLVE and is GA-based in its algorithm that considers effort and risk of requirements. Evolve* was also another version that was developed as hybrid model to address the problem of deciding which requirements should be assigned to which releases. S-Evolve* was introduced which considers the functionality and characteristics of the existing system as core knowledge when making release planning decisions. F-Evolve* on based on the financial aspects which can discuss which requirements can make the highest returns within the short period of development time. Evolutionary EVOLVE+ is an extension of hybrid approach that adds soft constraints and objective of RP to decision making process that were ignored in all previous EVEOLV approaches. It is widely accepted and validated in from real case study highlighted in SLR [48].

Next Release Problem (NPR)

This model is uses heuristics to perform the release planning process. It uses an optimization technique that considers some selection parameters such as customer's value requirement cost.

Quality Performance Model (QUPER)

This model [48] is used in Industry and is developed based on existing method “cost-value approach”. QUPER develop release plans based on some quality requirements where most of the existing approaches don’t not consider quality aspect at this level for release planning.

Quality Improvement Paradigm (QIP)

This model [48] is performed through six steps and it is goal is to deliver the client requirement within a short time. It uses a genetic algorithm computational through an iterative cycle of release planning process. It learns from the previous releases data for the improvements in future releases.

Incremental Funding method: This model is financial based approach that is designed to maximize returns through delivering functionality in 'chunks' of customer valued features. It is useful to analyze costs and estimate revenues of some periods

An Optimization technique for RP (AOTRP)

It models the revenues against available resources in each period. It uses an Integer Linear computation technique [48].

Fuzzy Model for dependence constraints in RP (FMDCRP)

This model handles the uncertainty of data using fuzzy logic [47]. It identifies structural dependencies between requirements

Post Release analysis of requirements Selection Quality (PARSEQ)

This model uses the backward analysis scenario. The quality of selected requirements in a previous release process is analyzed for proposing improvements. It is useful when the

requirements change many times and require the reprioritization process. PARSEQ is recognized as one of the most critical activities in market-driven software development [48].

2.5 Requirements Prioritization Parameters (factors)

Requirements prioritization process is trying to determine the different degrees of priority. Requirement feature has different relevant attributes/parameters such as risk, cost, complexity, time constraints, dependencies, scalability, contradictory, penalty, volatility, resources, speed, value, effort, approach type, result type, size of requirements, granularity, number of comparisons, structure, customer importance, strategic importance, expert biases, provision of change of requirements, empirical validation, ease to use, support for consistency, sales impact, customer satisfaction, marketing, strategic and integrity that must be considered during the process of prioritization. These factors contribute to the conclusion.

The prioritization practice could be used as planning process for different reasons such as selecting an ordered set of software requirements, scoping the project against some constraints such as schedule, budget, resources, time to market, and quality, estimating customer's satisfaction expectations, getting a technical advantage and optimize market opportunity, and establishing a relative importance of each requirement for software added value. It could be performed based on three several aspects: business, technical, and client each consists of set of properties considered as factors that contribute to the measurements process of the prioritization practice.

The following is most commonly explored properties in academic research work [38]:

2.5.1 Business Aspect:

- COST-BENEFIT

Cost-to-benefit is widely used in release planning evaluation process by software solutions providers targeting an increase in the market volume of a solution within a short period of time. This business aspect is used to determine the options that provide the best approach to achieve benefits while make some budget savings. It also defines a systematic process for calculating and comparing benefits and costs of during the decision-making process,

- VALUE

With the increased focus on value criterion in software development that could reveal the importance of this factor in requirements selection process. Typically, there are different and conflicting priorities among different groups of stakeholders. Value is an independent parameter where all stakeholders can reach to an agreement with no conflict. The influence of a required feature towards the best of software functions and business operations is measured under the business added value. The business value is considered as an element that has different forms such as business operation value, customer value, supplier value.

- RESOURCES

This is the most important aspect parameter of a software development management, this parameter should be examined from different perspective (time, skillset, and infrastructure). Resources are always limited in any project and

controlled. The time is defined as how much time can be given from a manpower recourse to participate in the development of a selected software requirement feature.

- MARKET VOLUME

This parameter gives a sign about the concern and interests for a solution provider to consider the leading position among other solution providers by implementing new innovative and unique ideas to be a pioneer in a specialized field that led to an increase on the market volume. It also gives a sign of the importance level in maintaining the company reputation in software market.

- Risk

A risk parameter is an incident that may occur and cause unexpected outcomes. The outcome may have a positive or a negative effect. A positive effect is an opportunity, while a negative effect may lead to downfall in business/dissatisfaction to a customer. A simple risk may consist of the following attributes:

- Description of risk: A one- or two-line overview of the risk. It should be precise one.
- Likelihood: Estimated probability of occurrence of the risk.
- Severity: The severity of the risk is assessed based on impact of the undesired outcome.
- Priority: This could be either given an independent value or set as a product of likelihood and severity. A high-severity risk with a high

likelihood should receive more importance than a high-severity risk with a low likelihood.

- Action: The response defined to manage/control the risk.

2.5.2 Technical Aspect:

- Volatility

One of the challenges in software development plan is requirement changes during the development phase. Usually, the involvement of stakeholders has a direct or indirect influence on the system requirements. Moreover, there is usually no formal way to describe the features. Requirements has been reported as one of the main factors causing the software deliverables to be delayed [31].

- Complexity

Requirement complexity of software development effort is a challenge for every software project, due to its strong impact on cost, schedule, functionality and quality of the software to be developed. There are several factors to be consider when examining the requirements.

- Dependency

Understanding the effect of requirement dependencies to software engineering activities is useful especially for requirement prioritization practice and release planning. Karlsson et al. introduced the dependency types to prioritize requirements [17]. Many dependency types have been proposed in various dependency models over years and most of them have different levels of abstraction and different criteria for categorization. This has increased the difficulty for evaluation. Among

these models, there are two representative generic requirement dependency models. The dependency model proposed by Pohl [29] was based on a survey of over thirty publications around requirements engineering (Figure 2-1).

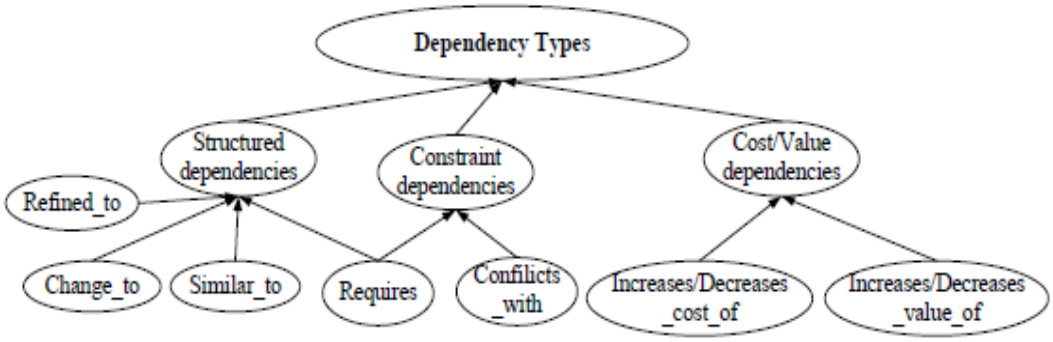


Figure 2-1: P-model for requirement dependency

- Productivity

Estimating how much time an employee/developer need to complete a task (analyzing/coding) is not an easy job [8]. There are many factors to be considered such as the skills and experience of the employee, the resources available and the difficulty of the task itself.

- Testing effort

Testing scenarios must be considered as factor to measure when measuring the complexity of newly requested requirement features.

- Maintenance effort

This type of effort implies the required time to support the requirements during the bug fix and defects. Maintenance effort are aided by understanding what happens to software over time. A software development continues to evolve over time. As they evolve, they grow more complex unless some action such as code refactoring is taken to reduce the complexity.

2.5.3 Client Aspect:

- Urgency

This aspect describes the criticality of requirement. It addresses the degree of satisfaction from the individual stakeholder perspective. It also addresses the time-to-market aspect, maybe, to reflect market needs and competitor analysis information. This factor measures the need of these requirements during the utilization of a software product.

- Importance

The client subjectively measures requirement importance when surveying set of requirements together.

- Stakeholder involvement

In most cases, stakeholders are not sufficiently involved in the planning process. This is especially true for the final users of the system. Often, stakeholders are unsure why certain plans were suggested. In the case of conflicting priorities, knowing the details of compromises and why they were made would be useful. All these issues add to the complexity of the problem at hand and if not handled properly, they create a huge possibility for project failures.

CHAPTER 3: APPLIED RESEARCH PROCESSES USING GROUNDED THEORY APPROACH

Background

Grounded theory approach was developed in the School of Nursing, University of California San Francisco by sociologists Glaser and Strauss [39]. It is defined as a discovery method that provides us with guidelines of building a theory from data collected from case studies, surveys, or literature and systematically obtained for a certain research practice.

GT is usually applied when there is no clear research problem [40].it is an inductive process in nature, which means that we as researchers have no clue to prove or disprove and relies on the concept of 'constant comparison', a process in which we constantly compare instances of data that we have named as a specific category with other instances of data, to see if these categories fit and are workable.

GT involves a progressive identification and integration of categories from data. GT has been recognized as practical research method approach in many science disciplines

especially in the medical field [41], and management [42]. GT studies often collect the data from the interview, documents analysis, and practices observation.

Parry [42] recommended the usage of GT as a qualitative research method to know:

- How to start a research (identifying area of interest, avoiding theoretical preconceptions and using theoretical sensitivity)
- How to do it (through analytical procedures and sampling strategies)
- How to stop (when theoretical saturation is reached)

Denzin et al. [29] lists seven different versions of GT methods, he did not specify the differences between all of them but, there were three main versions of GT [12] have been widely acknowledged and adopted by the researchers:

1- Glaser's GT (classic);

2- Straussian GT;

3- Charmaz's GT.

In classic GT, theory consists of concepts that are related to each other's from the definition or function points of view, which provides self-explanation and prediction to its content. Charmaz's GT focuses on the understanding of the collected data and interpretations, and producing a theory as result of the researcher's analysis [29].

In this research, we used the classic version of GT [12] and structured the documentation in two sections. First section provides an overview of the adopted and applied grounded theory guidelines. Second section, we provide a worked research study of a grounded theory as a methodology of an empirical study we conduct to come up with a theory or a

product named requirement characterization framework of requirement prioritization practice.

Grounded theory experts [44] recommend minimizing the literature review practice to encourage broadmindedness in the research and prevent the researcher from validating his/her findings against the existing theories or establishing concepts.

There is an increasing adoption in the field of software engineering studies to consider the social aspects in software development project [12]. SE researchers have subsequently adopted various research approaches from the social sciences. It is important that qualitative research methods are now adopted in SE research works [41] to identify the influence of people behavior on SE process. Grounded theory is an example of those social sciences research method that got an enough acceptance in SE.

We performed a quick search in Scopus to look at the statistics of GT studies employed in computer science field during the last two decades (1997- April 2017). It is almost 3,141 research works. Table 3-1 and Figure 3-1 illustrate this information.

The gradual increase of GT-based could reveal two things. First, most of SE research concerns are behavioral related practice. Thus, research uses GT approach as method to study the root causes of an issue that is subjectively reasoned to the way of people thinking practice. Second, it could also reveal the simplicity of GT method to understand the concepts and classification of a certain phenomenon make that interest among the researcher.

In this research, we applied GT–based study to explore all aspects of the RP challenges and extract some concept and categories, then put a consideration in each case of that discovered concepts when building a theory or a solution.

Year	Number of Publications
1997	18
1998	13
1999	17
2000	25
2001	34
2002	35
2003	54
2004	54
2005	95
2006	125
2007	176
2008	213
2009	240
2010	258
2011	291
2012	277
2013	268
2014	286
2015	302
2016	299
2017	61

Table 3-1: GT-based research publications in computer science (1997- April 2017)

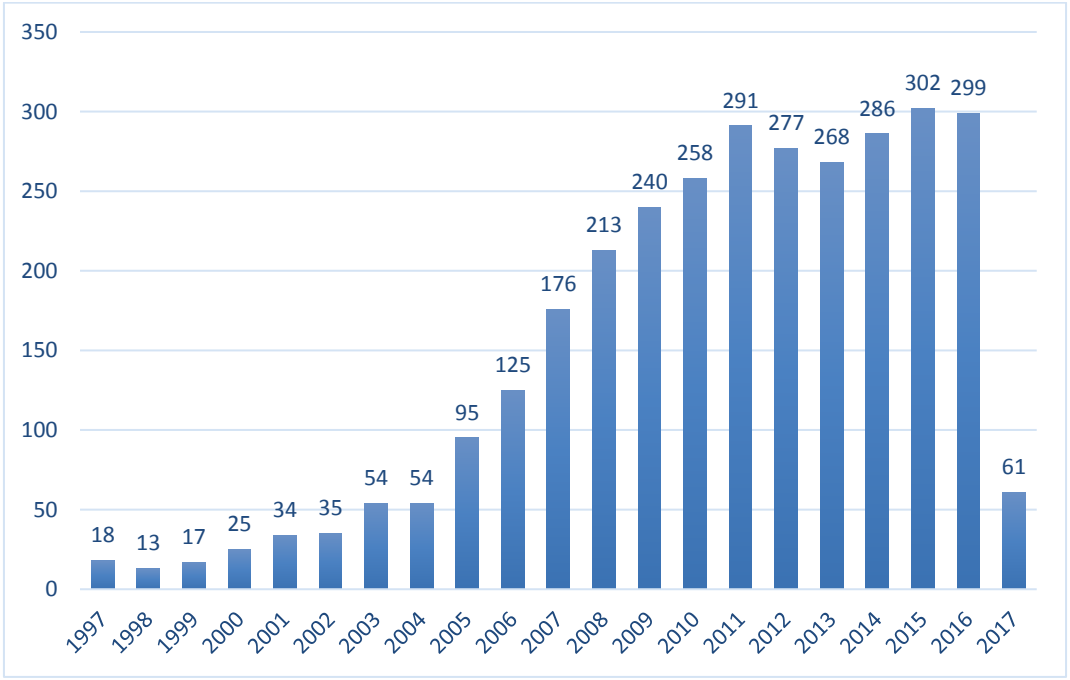


Figure 3-1: Growth of GT-based publications in computer science

Researchers have recently shared their experiences of applying GT by indicating its usefulness as a research method [51]. Most of them reported it's important because it produces constant evidence that supports the contribution to the field of a study. In practice, it is essential that researchers should design and report their findings of the studies inductively.

The motivation of using GT as research method in our study originates in the idea that software solution providers are not following 'best practice' in requirements prioritization for a reliable release planning process. On this basis, we initially set out a research question: *Why software solution providers are not using a standard RP practice?* to create a rich, explanatory theory of that software engineering practice and to clarify the transition from a subjective prioritization practice of software release planning methodologies to a characterization based method.

Since GT is a systematic analysis approach that facilitates the creation of a theory based on a study of a certain phenomenon or practice, a workflow concept was introduced by Glasser [40] to setup clear steps; Figure 3-2 illustrate the workflow steps.

Each GT workflow step has its own source of data or interpretations called *artifacts* and at the same time it works as source of data for the next step in the workflow.

The Research question: GT-based research initially requires some research questions to put the study effort focused on a specific phenomenon or concerns [12]. The question helps to identify the phenomenon of interest and its causes and implications. The GT initial research question should be open-ended not 'yes/no' answers to give a chance of collecting more

detailed information. It should also give a direction of researcher towards action and process as well as never imply making a conclusion derived from existing theories.

Data collection: GT has the flexibility to define the source of data channels it is compatible any form data collection techniques Semi-structured interviewing, observations, focus groups, existing texts and documents can also be identified as source of data for GT analysis.

Theoretical sampling: The researcher discovers different data types based on some gaps from the samples and seek more concepts to be explored. Theoretical sampling is non-deterministic as compared to the conventional sampling methods [12].

Coding: It is the initial step to establish uncertain relationships between categories through the data analysis. Coding process help the researcher to identify ways in which categories may be linked with one another at later stage.

Theoretical sensitivity: it gives the ability to extract concepts, and to establish relationships between concepts. It is one of the core process in GT method.

Memoing: Notes, diagrams, are all type of memoing is used to describe categories as they emerge, describe properties and relationship [12]. These memos are the most important step in theory generation as Glaser's said, "if the researcher skips this stage, he is not doing grounded theory" [12].

Sorting: It is a process that enables the interface between the memos and the concepts to find the suitable categories. It is very important and can't be skipped.

Theoretical saturation: Theoretical saturation refers to the stage in which a theory's components are fully supported and new data does not add much values to the body of the analysis process [12]. Grounded theory is unlike most other research methods. The researcher moves back and forth to 'ground' the analysis in the data.

Constant comparison: Data, memos, codes and categories are constantly compared in categories to ensure focusing on finding the differences *within* a category to identify any emerging subcategories. With this process, the full complexity and diversity of the data can be recognized, and any homogenizing impulse can be counteracted

Categories: grouping together of instances (events, processes, occurrences) that share central features or characteristics with one another. It can be at a low level of abstraction

Making a theory: the conclusion of studied phenomena will create the concept and categories in a form of highlighting the issues and causes.

GT Process	Artifact
Research question	Extracted as reasoning question of an issue
Data collection	Identify the input data channels and data type
Theoretical sampling	Select representative samples
Coding	Collect samples' analysis
Theoretical saturation	Capture more details
Constant comparisons	Merge findings
Build theory	Make the conclusion and produce a product
Theoretical coding	Test a theory from different point of view

Table 3-2: GT Process vs Artifacts

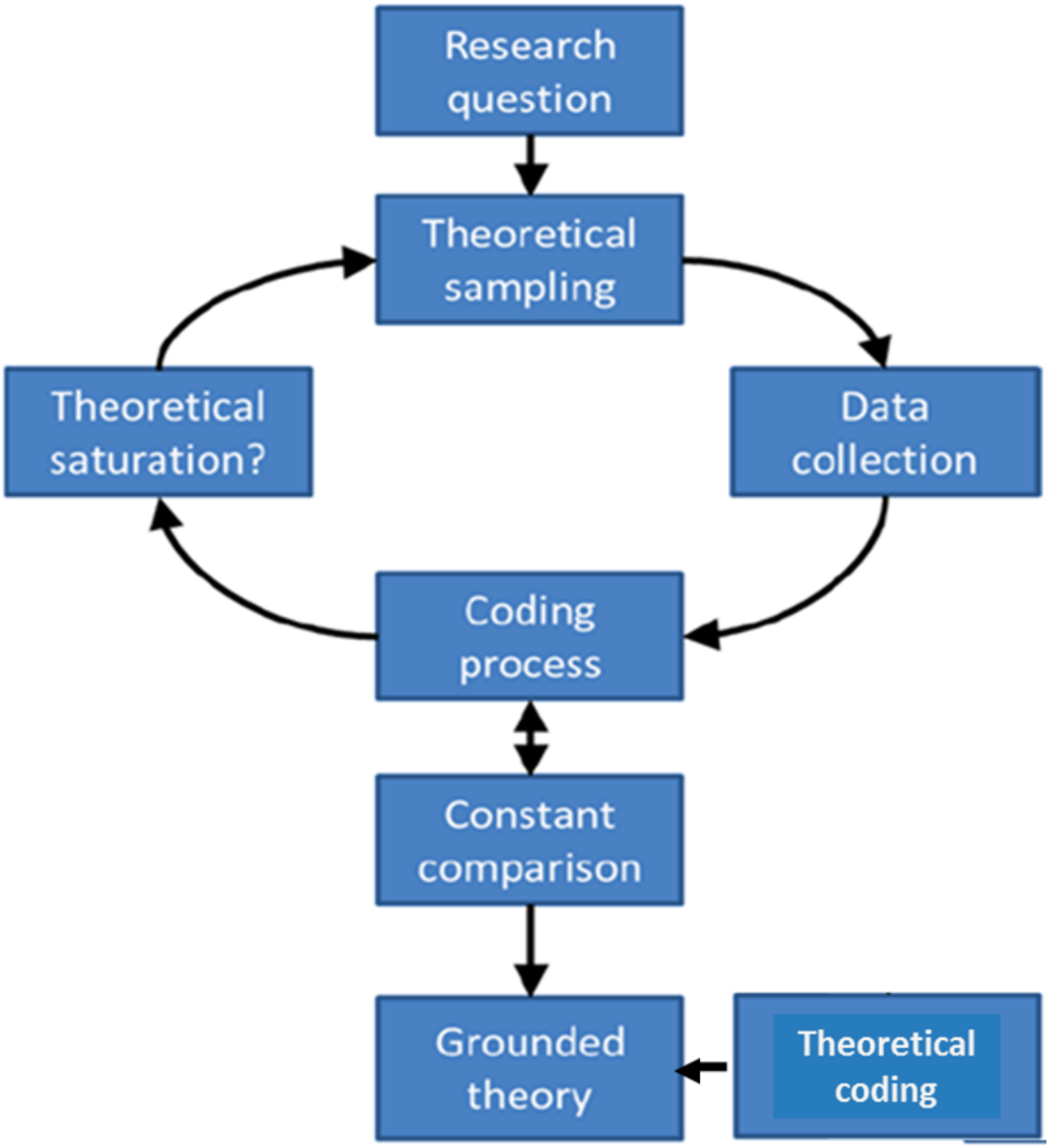


Figure 3-2: Grounded Theory Processes

3.1 Study Map

The study was conducted in a form of three different scenarios: interview, observations and previous document analysis. Each scenario has its own GT nature of process treatment as shown below in GT process diagram:

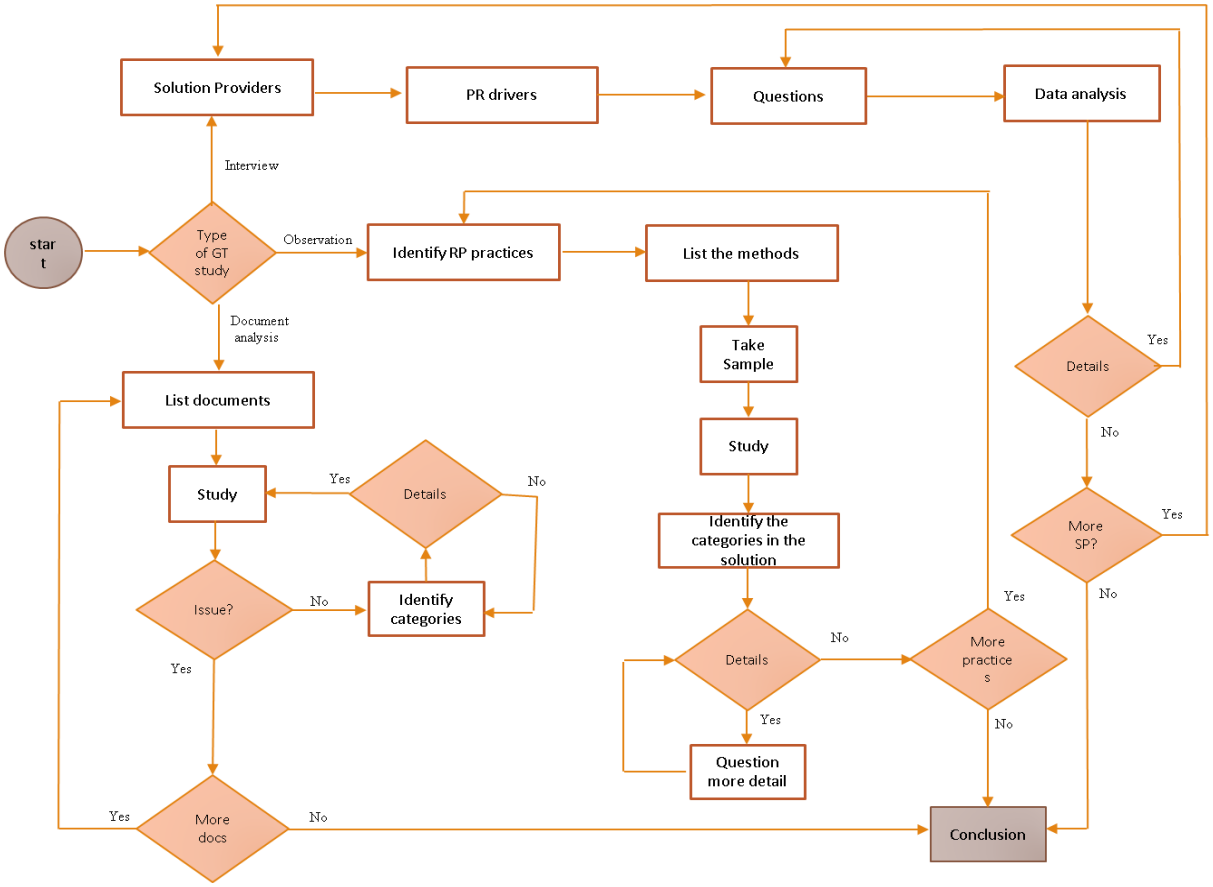


Figure 3-3 Applied Grounded Theory Study Map

3.1.1 Research Questions

A review of some studies indicates that there is no common practice for RP [28]. Also, there is no standard practice that has gained enough acceptances among software development companies. In fact, the inconsistency issue in release planning explored in the research studies stated that issue is only the outstanding challenges of release planning that has not thoroughly addressed. Thus, building a standard process to address that concern involves a lot of process to be considered in organizational practices, process, and people

.

Therefore, to ensure the collection of all the required data from what we specifically focus on and what we want to investigate about. Below are the employed research questions:

- **RQ1**: How do software companies prioritize the software requirements and plan the releases?
- **RQ2**: What are the priority decisions factors?
- **RQ3**: What are the issues of the RP practice?
- **RQ4**: What are the drivers of the applied prioritization method?
- **RQ5**: When requirements prioritization is carried out?

With these main questions, we conducted an empirical study using GT method to manage the focus of the study not to use them for interview.

Research question	Topic
RQ1	Current requirements prioritization practices in the companies
RQ2	Factors that have, or should have, an effect on priority decisions
RQ3	Problems that companies have with their current practices
RQ4	Sources for priority information
RQ5	Development phases in which requirements are prioritized

Table 3-3: GT study Focused Topic

3.2 Data Collection

There is a lot to be learned just by observing the practice with the organization. Data collection step is the main source of the information. The initial stage of data collection depends largely on a general subject or problem area, which is based on the proposed research questions' perspective of the subject area. We began identifying some key concepts and features which we will research about. This gives us a foundation for the research. We adopted the theoretically sensitive from that beginning so that a theory can be conceptualized and formulated as it emerges from the data we collect.

3.2.1 Interview: Software Industry Practices

Three participants from three different IT solution providers in Oil and Gas industry were interviewed about their applied RP practices (*Schlumberger-SIS, Halliburton-Landmark, and Paradigm*) during their software development lifecycles. The reason of this data collection step is to collect, understand, and classify the main drivers of the RP practice from the industry. The interview was conducted in a form of an open-end question to know the objectives led the solution provider to consider different when performing RP practices (see Table:3-3).In order to follow Glaser's advice [39] regarding the interviews questions and how we continually change them to specifically focus on the concerns which seem central for the concepts, categories, and theory, we limited the interview to one on general question to several reasons:

- Participant can come up with their own clarification questions which will help us to trigger more some other directions of the discussion

- Introducing many questions where some of them might be applicable to some participants and some aren't applicable thus, this gives an opportunity to have guessed answers which hold back the induction of concepts and categories.
- Make simple and attractive to participant. All participants we interviewed has been requested by email to send use a feedback about their experience of applying RP practice. This feedback could be interpreted as a response in a form of technical report with details and examples, or just a simple paragraph especially if we are considering one of the important client of their software client projects.

Table:3-4 shows the number of Participants and their considered aspects. Table:3-5 describes the participants job description and their relation to prioritization practice

IT solution providers	**#Participants**	**What are the main drivers for RP practice and release planning process?**		
		Business	**Technical**	**Client**
A	2	Cost-to-benefit, value (1)	Recourse (3)	Needed (2)
B	1	Cost-to-benefit (2)	-	Need and applicable to others (2) Urgent (1)
C	1	Value	Recourse (1)	Importance (2)

Table 3-4: RP perspective (interviewed)

Company	**Participant's job description and his relation to prioritization**
A	**Product champion**: Main task is to elicit and prioritize requirements.
A	**Software development Project Leader**: R&D unit. Good experience in requirements prioritization.
B	**Project manager: Collects** information about markets and writes requirements specification, good experience in requirements prioritization
C	**Product development engineer**: implement a requirement management tool to the organization and participated in requirement prioritization practices.

Table 3-5: Participants

3.2.2 Observation: Document Analysis

Similarly, we identified four software development projects and extracted some information related to the prioritization practices and the release planning documents. This type of scenario was initiated to find the used factors of the used RP practices and contain the introduced deficiencies. The review studies of these documents will also meet the answer requirements of RQ2 and provide some information about for the impact of the frequent priority changes.

In-house Software development	inconsistency issue%	Release date	Actual deployment	Main Reasons
SDLC-Proj-1	20%	2005	End 2005	Few requirements, scientific domain
SDLC-Proj-2	34%	2007	2008	Complexity, level of requirement abstraction
SDLC-Proj-3	11%	2009	2009	Few requirements, frequent changes
SDLC-Proj-4	65%	2014	2016	huge # of requirements, requirements changes, limited manpower

Table 3-6: SDLC documents analysis (observation)

3.2.3 Review: Academic Literature

Literature review was the active acquisition of data from a primary source, it involves the recording of data/information based on different type of methodology perspectives that could be subjective or qualitative.

Glaser [40] had much of the prior background reading which provides the models to help make sense of the data. He recommends reading widely while avoiding the literature most

closely related to what you are researching. That avoidance to minimize constraining the coding and memoing process. Therefore, Literature studies [2], [9],[16] provide us with level of domain expertise and latest advancements and give us a foundation to meet the study requirements and the proposed research question RQ1.

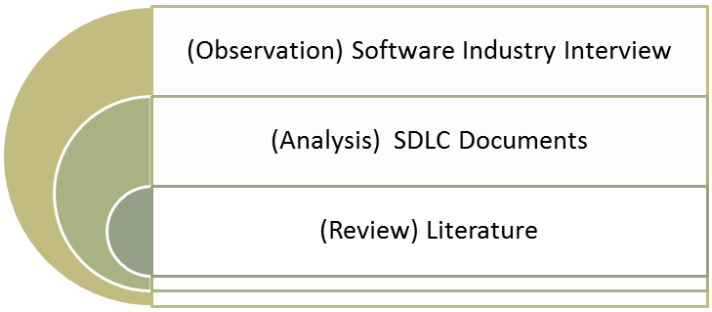


Figure 3-4 the size of GT study scenarios

3.3 Analysis

3.3.1 Theoretical Sampling

Theoretical sampling is a process of data collection for generating theory whereby we jointly collect codes and analyses data and decides what data to collect next and where to find them.

The initial data sampling starts in each type of scenario separately. In the first scenario, we had a prior background reading which provides the models to help make sense of the data. In other words, theoretical sampling is not about sampling list of tasks/processes but sampling concepts. Thus, we went to the places, persons, and situations that will provide information about the concepts you want to learn more about. There are more chances to explore more and more details later one which will meet answering the rest of research questions and participate in identifying the categories to add more sample in such a way that further increase of diversity is achieved in useful ways.

3.3.2 Coding Process

Coding process is considered as the core of the GT research method that takes care of analyzing all collected data have a systematic data process activities. It contributes to theoretical sensitivity, which gives the ability to understand the data's important elements and how they contribute to the theory. According to Strauss and Corbin [17] said, "the theory that is derived from the data is more likely to resemble what is actually going on than if it were assembled from putting together a series of concepts based on experience or through speculation". Table 3-7: describe the applied open coding process

In this coding process, we addressed one of the challenges of understanding the data analysis results from the abstract point of view. Thus, applied three different type of coding process to answer the research questions (RQ1, RQ2, RQ3, RQ4, RQ5) by identifying the main properties of the answers

- Open coding: we read through our data several times and then start to create labels for group of data that summarize what we see happening (not based on existing theory –based on the meaning that emerges from the data).
- Axial coding: we identify the relationships among the open codes. What are the connections among the codes?
- Selective coding: Figure out the core variable that includes all the data. Then reread the transcripts and selectively code any data that relates to the core variable you identified. Table 3-5 below describe the applied open code process in this study:

Focused topic	Participants Quotes	Initial coding	Theoretical coding
RQ1	"based on business demands" "customer satisfaction"	No standard method Requirements prioritization is an ambiguous concept	Standardization
	"based on our goal", "based on an objective"	One-dimensional	Needs to be comprehensive
	"We try to judge costs in the early phases of development. We have no formal method for that. " of course, value is the main driver"	Benefit-based	Cost-to-benefit
RQ2	"it is not considered"	Difficult for MCDM	Practical, simple
	"no enough data for DM process"	The priority of a requirement is based on some selected factors	No common practices serving different purposes
	"can't be by mulita-based DM process, large scale of requirement"	Developers do not know enough about customer preferences	Scalability issue
RQ3	"postpone the implementation"	Prioritization practices are informal and dependent on individuals	Should be easy to use
	"More data is missing, result requirements to be dropped"	Require some efforts	Not measuring the technical complexity of that model
	"full understanding of requirement features"	Constrained after requirement elicitations	Consistency issue
RQ4	"we usually consider one aspect", "Our local areas have the same problem as we have. How to know what is truly important to customers?"	Business, technical, client	Impacting indirectly the release planning: consistency issue
RQ5	"Client put the requirements, we analyze it in terms of functionalities and complexity" "We have a person who knows what it takes in the way of resources to implement the requirement and a person who knows how much effect it has on business. It is just a mutual discussion."	Requirements are prioritized in many phases	More consideration is required to the level of abstraction and dependency.

Table 3-7: Applied open code process

3.3.3 Concepts and Core Categories

In this GT research study, we began coding/data analysis in each iteration during the sampling step to identify concepts, grouping them to categories, identifying the categories´ properties and dimensions; in every practice. Figure 3-4 below identifies the common explored concepts and identifies the main challenges (categories)

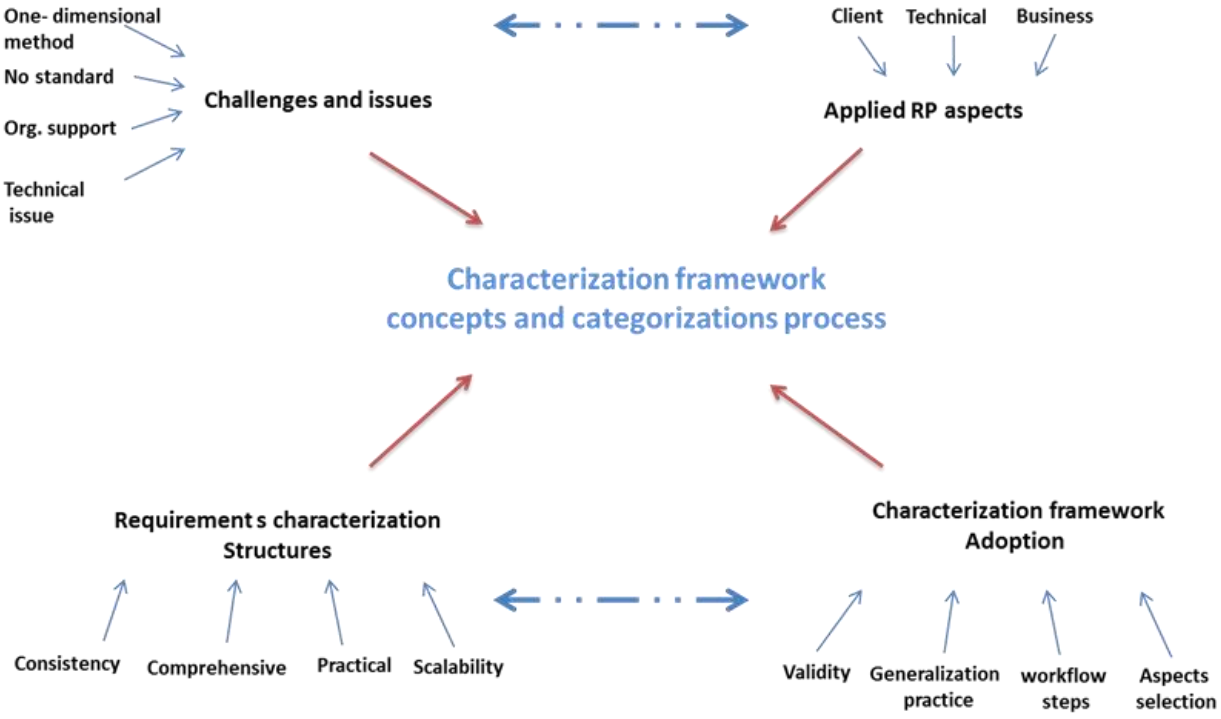


Figure 3-5: Emergence of RP practice categories and adoption of characterization concepts

RP technique	Scalable	Comprehensive	Consistence	Practical
Numerical assignment	Yes	No	Yes	Yes
Ranking	No	No	Yes	Yes
Priority groups	No	-	No	Yes
Game planning	No	No	Yes	Yes
Requirements triage	No	No	Yes	No
TopTen	No	No	Yes	Yes
QFD	Yes	-	-	No
Cumulative voting	No	No	Yes	Yes
VOP	Yes	-	-	-
TOPSIS	Yes	-	-	-
AHP	No	No	Yes	-
CaseBase	Yes	-	-	-
EVOLVE	Yes	-	-	-
IGA	Yes	-	-	-
RUPA	Yes	-	-	-
Minimal spanning tree	No	No	Yes	Yes
HCV	Yes	No	Yes	-
Hierarchy AHP	Yes	-	Yes	-
Bubble sort	Yes	No	Yes	Yes

Table 3-8: RP practice challenges' categories

In Table 3-6, we identified the challenges of each RP practice and categorized its function in terms of the features such as comprehensiveness, consistency, practicality, and simplicity for both scenarios in this applied GT study. Thus, this data analysis step gives us a chance for exploring the challenges of all categories and enriches the research study for further details and technical concerns to be considered when developing the solution.

3.3.4 Theoretical Memo/Noting

After each interview and document review ideas and concepts started to be noted during the data collection activity. This process is known as theoretical memo [39] which are notes for hypothesis about a category or property, and particularly about relationships.

In fact, the emerging codes, concepts, and categories and relationships give some ideas captured as a memo and then, we collected more data for the memos. In effect, memoing task adds usually relationships (link the categories to each other).

3.3.5 Sorting

Sorting is the compilation process of the emergent categories with its related concepts and links. It started when all codes reached the saturation level, and data collection was almost finished. In our research sorting process was conducted by the grouping based on the similar categories or properties. Then arrange the groups to reflect their relationships. Applying this procedure helped in structuring and outlining the required RP solution requirements. It also shows how we structure the study report to communicate the theory to others.

3.3.6 Study Report

First, the study revealed some important findings like what have been explored in the literature studies [34] [50] with respect to the challenges and concerns of RP practices and their aspects categories. This confirm the agreement of the findings in both source of the problem investigations. Apart from these similar categories, we extracted some other categories related to the structural attributes of RP method and its adoption process in which it has been identified as core categories of the requirements characterization framework concept.

This study also, revealed some important implications for acceptance. The results explained the realization of a solution framework and its implementation which is identified as the core of RP practice. Software companies must understand the importance

of incorporating all aspects of a requirements through a characterization step and before starting the prioritization process.

The proposed RP practice is a combination of two processes which mostly focuses on people and process. In fact, that makes the characterization method promising compared to the traditional RP practices. Also, the study supported the previous research studies that highlighted the advantage of considering a several aspects to achieve a reliable prioritization practice.

This study also put some constraints for a successful framework to be comprehensive, practical, scalable and iterative. These characteristics represent the main features of an effective practice. Each of these characteristics emphasizes a critical aspect of requirement characterization. Furthermore, since achieving business benefits is the main aim of the framework, the characterization framework should be value-oriented .At the same time, the ability to define several properties (factors) that assists in its implementation is important.

The inter-relationship between two selected properties of requirement characterization and their contribution direction was another finding of this study. Although there is no standard practice for requirements characterization adoption, the emerged steps, seem to be sufficiently useful when it introduced the competition scenario and made applicable to all discovered categories regardless of their relationship and property type.

The results showed that requirements characterization can be applied in any release planning process if it satisfies the main characteristics and steps of that framework. Furthermore, such a framework is consistent with the characterization method. Software

solution provider can prepare an action plan based on this approach and apply it during the release planning process. Such a framework that is easy to use and practical can be applied for even small number of software requirements.

3.3.7 Theory Building

Theory building or theoretical coding is the final step of this GT application and was applied to identify the relationships between the core categories, and form the hypothesis that create at the end a theory, solution, or product [39].

In this application study, a solution that adopts addressing the concerns reveled in a form of category or link has been considered as solution components.

Therefore, a discovered theory makes the specifications of the final "product" of a proposed solution framework. There are different structures of theories, called "theoretical coding family" [39]. One of the most popular codes in this family is about the process "|*concept"*. We used this coding family to formulate the competition-based and characterization concepts. Each coding family (concept) demonstrates category in the form of *elements*, *types*, or *properties*.

As part of theory building process in finding the links and connection which revel ability to enable the emerged categories which is explained in a form of process coding approach. Figure 3-5 illustrates the competition-based and characterization concepts identified the core categories and how links and connection described the transition steps that makes the comprehensive solution framework to address the stated concerns of requirements prioritization practices.

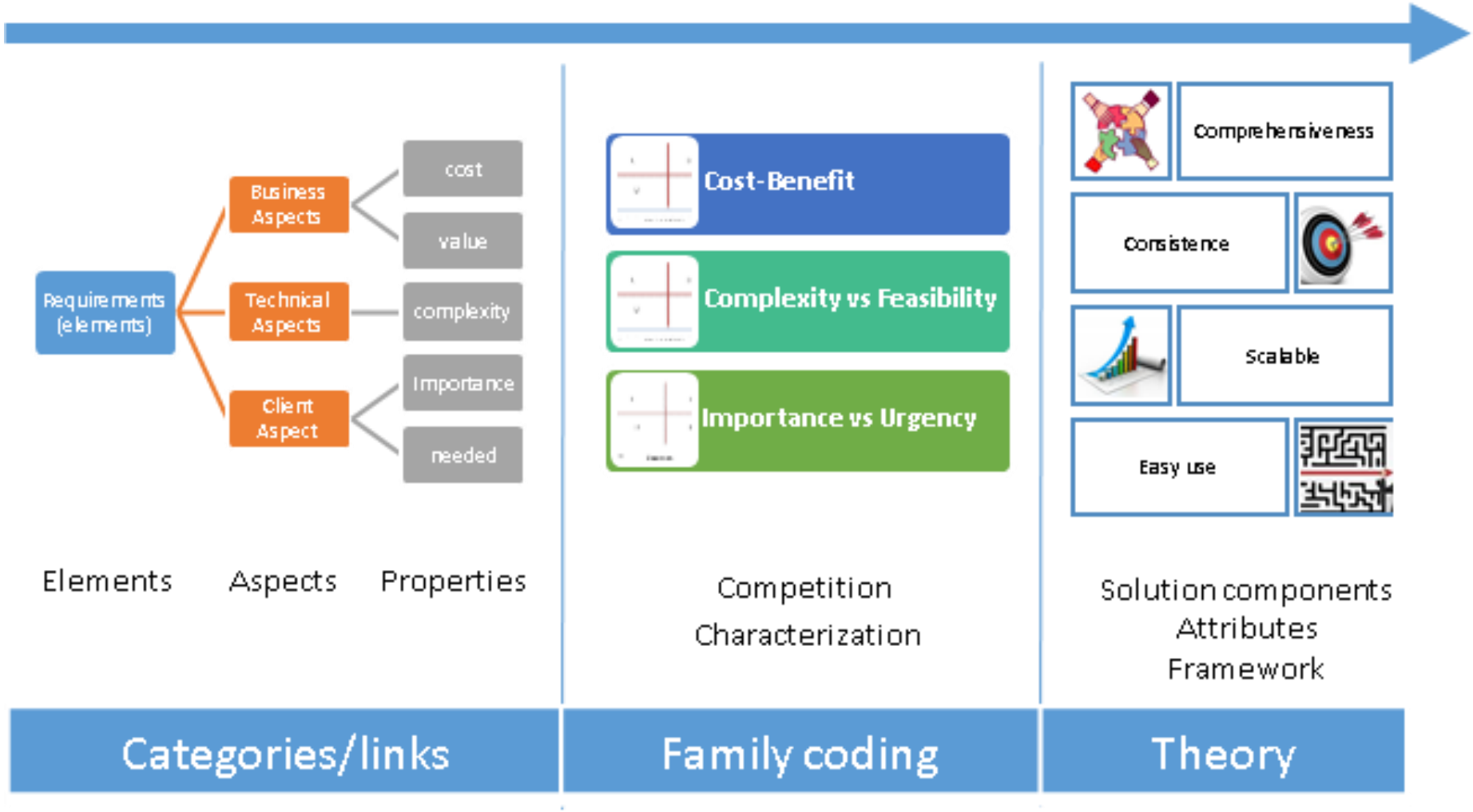


Figure 3-6: Transition steps of the characterization framework

3.4 Results

3.4.1 Structural attributes of the characterization framework

During the theoretical coding step, we identified the emerged categories and identify the most important attributes that can be useful for RP practice. We believed considering these attributes as fundamental features of the RP practice, makes it less costly in terms of the required time and effort in any future demands, and more effective approach to consider at the beginning of release planning. In below section, we explored the impact of each attributes as result of the requirement characterization framework.

3.4.1.1 Comprehensiveness

This is the most important attributes which is used during the evaluation process to assess all aspects of a software requirements prioritization on a continuous basis throughout the development lifecycle.

All the solution providers' participants emphasized that requirements prioritization practice should be a multidimensional evaluation process. Considering the business, technical, client aspects is the most important factors which lead to successful release planning. Requirements characterization framework helps solutions providers to achieve, as much as possible, of these aspects. Of course, defining and selecting appropriate metrics for measuring any aspects are significant factors in designing an effective and in-depth characterization framework.

Solution providers need to define properties before starting RP practice and requirements characterization framework must consider such properties values in its workflow. The adoption task of this attribute should be a satisfied by considering a multi criteria decision making process to find all factors for the requirements characterization.

3.4.1.2 Consistency

The consistency attribute is the crucial part in declaring a successful release planning process. It is a one form of requirement validation process and its objective to check whether the delivered software requirement is satisfied. It also facilitates the changing tasks between various parts of the software requirements being collected, developed, and released. The requirements consistency attribute we discuss has several many aspects to be looked at such as interfaces, control parts, representations, relations with other requirements.

In data analysis, we identified that requirement volatility cases as the most important factor that could reveal indication about the inconsistency during requirement prioritization practice. A dynamic method to review periodically requirements changes and have reflected and acknowledged by software engineers, users, and other stakeholders to ensure that the limitation of consequences issues associated with consistency are detected and corrected. For example, adopting a multi-level process, like spiral method, could be useful to contain the consistency issue.

Therefore, software companies and organizations can use a simple and well-known design for release planning rather than using a complex prioritization framework to address the inconsistency issue. However, considering this simple design won’t have the same attributes

that the characterization framework is important and can help solutions' providers to avoid obstacles they might face during prioritizations practice.

3.4.1.3 Scalability

Prioritization process can sometimes be time consuming and difficult for its stakeholders. The process of comparing requirements to each other for ranking becomes complex as the number of requirements increase [38]. Hence, this process should be done at the time of eliciting and analyzing requirements when their number is quite small and it would be easy for analysts and stakeholders to interact with each other and agree on a set of requirements which need to be implemented while developing the system. Typically, the customers and developers may be doubtful towards this process because customers feel only the important requirements will be implemented and developers are afraid to admit it [35].

Therefore, scalability attribute is amid in GT study to manage the prioritization step of many requirements. Most of the studied approaches used the concept of the concept of pair-wise comparison that is time consuming and suffers from exponential growth as the number of requirements increases which defects the prioritization effort. In fact, data analysis showed that as soon as the number of requirements increases, scalability will limit severely the applicability in most of the explored RP techniques. In this regard, we need to adopt a kind of solution requirements that define a limit on the variation of the estimated requirements ordering under which new comparisons are no longer needed.

3.4.1.4 Practicality

A Practical approach is the last emerged structural attribute of the proposed characterization framework .it is basically defining a clear-cut step and of RP method to be easy to be understood and applied for any requirements set.

Data analysis stated the easy to use as one of the challenges most of the studied RP practices are not considering. Adopting this attribute is an opportunity to be addressed in the proposed framework. Most of IT solutions providers are not really interested to use any RP method that is complex or unclear.

3.4.2 Characterization framework components

In Section 4.3, several structural attributes were introduced to address the settings of the proposed solution. It revealed the appearance of several key components in requirements characterization framework and adoption process. The below section describes these components and provide a description of the adoption process.

i. Elements

Element is the core item in this characterization framework. It is the elicited software requirements. Each element should be defined in a form of types and levels. There must be a clear a description of a requirement that each stakeholder can understand and measure from business, client, and technical point of views.

ii. Aspects

Aspect is a classification type of a selected criteria that need be used as a property in the characterization framework. Example is the *risk and penalty* properties; both of them are classified under business aspect.

iii. Properties

This part is the actual quantification attribute to evaluate a requirement feature during the characterization workflow process.

iv. Competition method

It is the core method in characterization framework used to evaluate a software requirement against pairwise selected properties. The input of this method will be a requirement element and the grid properties to quantify each element from four different class defined in that grid

3.4.3 Framework adoption

The adaptation of requirement characterization framework is implemented as three main workflow processes to control the workflow of the prioritization practice Figure 3-7:

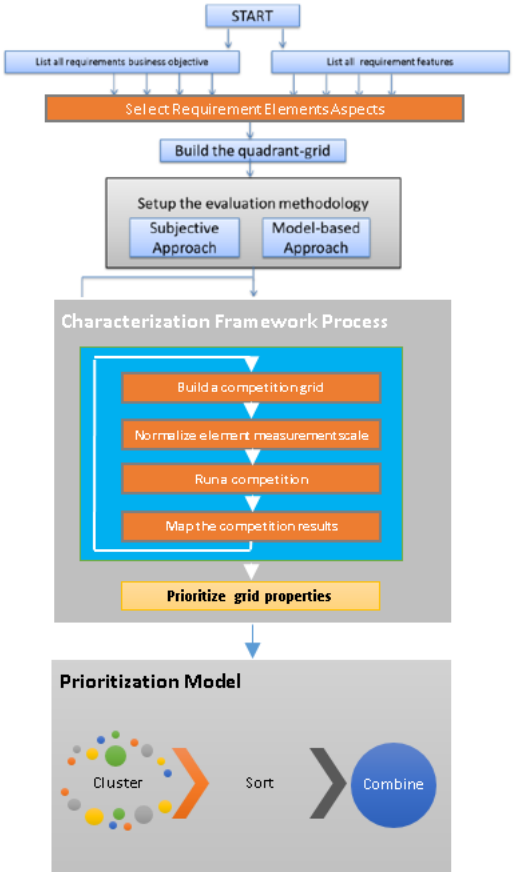


Figure 3-7: Requirement characterization framework adoption

- Requirement Initiation process:

In this process, requirements features are listed and grid properties are identified from the three aspects, business, client, and technical.

Understanding requirements details increases the possibility of project success. During the requirement analysis step, a requirement feature is quantified to users' expectations and technical details. We assume in this process that the requirement feature is in low-level of abstraction in which the complexity is measured in terms of size and structure categories. In size category, is code-based measurement for length or functionality factors. The challenge in this it cannot be asses it at early stages in the software development process. However, in structure category, it considers the software structure by means of flows, control, structures, and interaction as factors which can reveal some information about the dependency, infrastructure and integration of a requirement feature.

On the other part of this initiation process, List of properties (factors) that is selected to be used as a property that will be assigned with right value during the characterization process. Each property is subjectively assigned with a value of four categories calculated using the competition grid.

This final step in this process is which focus on building the strategic-quadrant grid by taking a pair of properties considering the *interrelated-compliance* condition to build the grid and map the order of class A, B, C, and C subjectively. Each class has an assigned value that characterize a requirement when it fits into it. Thus, a requirement *E* is evaluated on the basis how it response when it being examined from two points of properties/factors.

- Requirement characterization process:

This is the measure process in the framework, where we build the competition grid and assign a rank for class of the competition categories.

Building the competition gird requires a study of the inter-relationship between two properties. If there is no inter-relationship between them, then building the grid for those properties is ignored.

The purpose of this process is to figure out a concept of the contribution direction. The definition of the contribution direction is for high/low axis; if both properties axis types are logical have the same direction towards (undesirable or desirable) then, it won't contribute fully to result and the assignment of C and D categories will equal in their sorting. Figure 3-8 and Figure 3-9 illustrate the concept.

	COST	BENEFIT
HIGH	undesirable	desirable
LOW	desirable	undesirable

COST			
	HIGH	D	B
	LOW	C	A
		LOW	HIGH
		BENEFIT	

Figure 3-8: Full-contribution Inter-relationship

	COST	RISK
HIGH	undesirable	undesirable
LOW	desirable	desirable

COST			
	HIGH	B	D
	LOW	A	C
		LOW	HIGH
		RISK	

Figure 3-9: Partial-contribution Inter-relationship

- Requirement prioritization process:

The characterized requirements are produced as an output from the framework, then clustering, sorting and prioritization steps are performed to create the priority list.

Since the characterization process is completed and all requirements are characterized from different aspects, we started the prioritization process which simply consists of three modules as below:

Clustering: In this module, the requirements are put together based on the resulted values from the competition in each grid. This module-step is important to control the weighting factor by counting the frequency number for each class type and to minimize the human interference to the result and address the highlighted concern about the subjective weighting factor practiced in many proposed release planning models [55].

Sorting: we determine the importance of the element based on the resulted value from the clustering step. This is basically applying a similar concept of PageRank algorithm [55] used by google search engine to rank and sort the search resulted web pages based on the importance.

Ranking: It is the core and final module step in the prioritization process. It calculates rank of each requirement element considering its individual performance and it is weighting factor among set of prioritized requirements.

3.5 Research study validation

Glaser and Strauss [43] consider the reason of using GT to build a theory, not to validate it. In fact; they consider its validity is not a critical issue in GT. Therefore, other chances are giving to the researcher to apply it and verify. Glaser claims that an emerged grounded theory is grounded in data and in some respects already has been verified. "A grounded theory gets its concepts from the data; it does not bring ideas to force the data that need to be subsequently tested".

We considered Glaser's criteria [43] highlighted for evaluating the credibility of the emergent theory and its related categories that comes out of GT research efforts: *fit, workability, relevance, and modifiability.* We will discuss their implication of practice from the adaptation point of view. Table 3-9 GT summarize that.

Table 3-9 GT application study validation

Validation criteria	Definition	Practice
Fit	Focuses on codes, categories, and theory from data rather than researchers' thoughts of ideas. It emphasized that GT researchers must ignore their own perceptions and remain open to the data [43].	We found the diversity of that data collection makes it more supportive in building the final product in conducting the research questions, analyzing the document, and reviewing the literature.
Workability	Assess the integration of the core category and its main related categories. A theory should be able to give details of what is happening in the area under study and out a prediction of what will happen next. It shows how the progress of concerns/issues under study is continually resolved.	We introduced a reviewing process after each GT practice to review the progress, new findings and consider new induced issues/concerns.
Relevance	Test the focus of a grounded theory application towards the identified issues/concerns. It allows core problems and processes to emerge". GT researchers can assess workability and relevance criteria by asking the participants whether the emerged categories and theory relate to the main issues of the study [43].	We maintained the satisfaction of this criteria within the reviewing process of during data sampling and coding process
Modifiability	To measure the ability of the theory to be continually modified upon acknowledgment of new data. As Glaser states [43]."a category, which fits and works, is relevant and is subject to continual modification".	We meet this criteria by limiting the open-end question to only one. Assumed participants will enrich the discussion.

CHAPTER 4: PROPOSED REQUIREMENTS CHARACTERIZATION FRAMEWORK

Solution Requirements

The common highlighted concepts and categories explained in section 3.2.3 are the main pillars in the proposed solution including (*comprehensiveness*, *practicality*, *ease of use*, *and consistency*). There must be a mechanism to dynamically consider all those pillars (factors) within constructed framework solution.

The consistency attribute is modeled to make the framework solution aligned with release planning process which ultimately increase the reliability factor. The framework solution should also provide a type of scenario that makes it comprehensive to support the different aspects of RP practice where the added value, cost, needs, urgency and complexity of requirements features. In addition, the prioritization model should be practical and ease to use to address the common concerns [4]. Figure 4-1 shows the solution requirements and attributes.

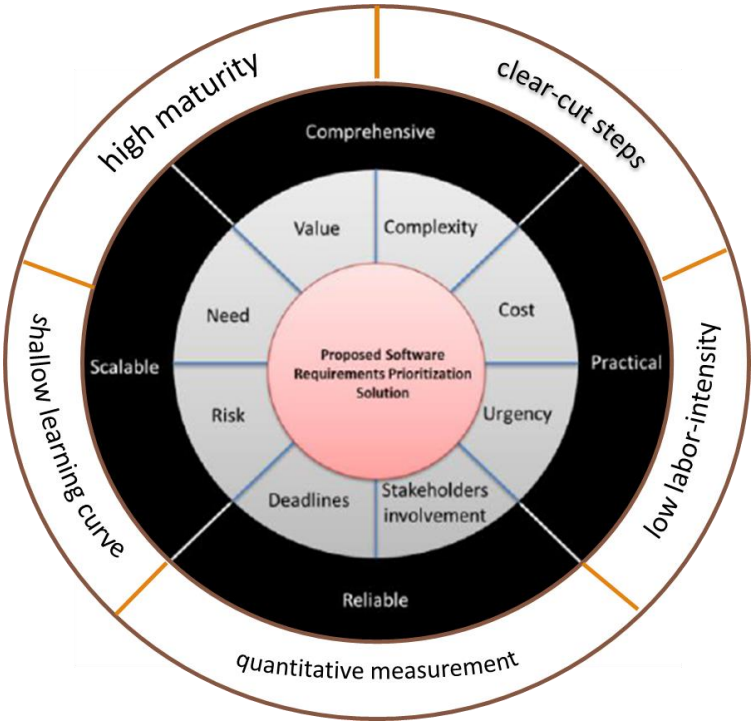


Figure 4-1: Solution components requirement

3.0 Solution Components

As explained in section 3.3.1, the comprehensiveness of the characterization framework can be addressed in such a way that considers all key aspects during the requirement prioritization practice. Thus, the proposed solution will serve all main characteristics and steps identified from GT study in the normal situation of RP practice. Furthermore, we conducted an extensive search process to explore all the applied evaluation methods that consider multi-criteria aspects in their decision-making process in which it collects more information about the items that need to be evaluated. We found a method applied in medical field called strategic-grid method [8] that can prioritize patients for medical treatment based on classification, grouping, or characterizing set-up in a form of a competition.

To understand the work steps of the proposed competition-based requirement prioritization model, there are some basic principle components such as strategic-grids [8] and competing properties will be explained in the coming sections.

4.0.1 Strategic-grids method (competition-grid)

Strategic grid approach is applied in medical-treatment field where medical personnel prioritize the patient based on {criticality vs treatment} into four categories:

- **DIE**: Treatment is low and the case is highly critical
- **SURVIVE**: Not enough treatment and not critical
- **GOOD CARE**: Enough treatment and not critical
- **SIGNIFICANT IMPROVEMENT**: High treatment and case is high critical.

Figure 4-2 shows an example of that treatment. A company can use the strategic-grid to classify their software requirements with other stakeholders as a preparation step to get the "big picture" and have enough knowledge for RP practice. It also provides an idea about the critical and near future development plan and efforts/resources allocation by focusing on addressing highly-prioritized requirement features first. This model becomes more useful when a company is limited in recourse capacity and wants to focus on areas that provide 'the biggest outcome' via a mechanism to take a thoughtful approach to achieve maximum results with limited resources.

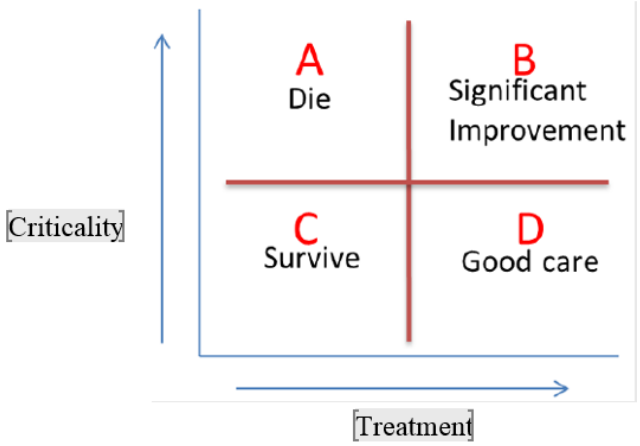


Figure 4-2: Strategic quadrant-grids for treatment classification

The patient treatment will be prioritized in the direction of a positive health support which we can assign A, B, C, and D as a rank for the prioritization purposes. The usage of this quadrant-grid in our design model will be to characterize set of requirements towards the requirement development for a software releases. To determine the selection factors when performing the requirements prioritization practice, simple and easy grid with four quadrants that each one has its assigned value can reflect the evaluation result from a competition performed between two interrelated properties.

In addition, the comprehensive attribute of the requirement prioritization process can be achieved by considering e.g. the cost, value, importance, deadlines, complexity, risk technological constraints, and quality constraints.

Therefore, a part of our model is a preparation step to collect and combine several properties that are required for competition purpose, which means two properties that could be implicitly related to each other e.g importance/urgency, need/feasibility, cost/value…etc.

The following steps show the guidelines of building characterization framework:

- Select properties – by choosing two properties that are relevant to the stakeholders/company property (e.g. 'importance/urgency, 'cost/impact', 'need/feasibility,' risk/penalty',etc.).
- Create quadrant-grid – Set up the four quadrants and assign the properties to each axis independently.
- Create arrows on the axes to indicate 'high' or 'low,' as shown in Figure 4-2.
- Assign quadrants matrix-value – assign (0.25, 0.5, 0.75, or 1) based on the degree of a *positive reaction/contribution* e.g in Table 4-1 show the class and the assigned score value. The reason of scale selection is to let all the input variables have the same

treatment in the model and the coefficients of a model are not scaled with respect to the units of the inputs

Class	Value
A	1
B	0.75
C	0.5
D	0.25

Table 4-1: Grid class type vs score value

Adopting this form of scoring system (low, high) could be debatable and exposed to a threat to validity case when it is applied with fixed scoring system. However, the objective of this evaluation method selection was not to capture the uncertainties from different stakeholder's responses but to find a distinct feature that can attribute each requirement *E*, eliminate the ambiguity of using the framework by assigning requirement *E to a distinct and well-defined grid* quadrant.

4.0.2 Grid properties binding

To build the competition grids, we need to identify the following:

I. Interrelated properties identification

In this identification process, we prepare the number of competition grids applied for the requirement elements considering the uncertainty cases to be excluded. These cases can be detected when there are two properties are same or contributing to the same direction.

The maximum number of competition grids is relatively calculated as follow:

$$CG = (gpn^2 - gpn)/2$$

Where

CG : Number of competition grids

gpn: Number of properties

II. Grid class mapping process

In this process, we assign a class (A, B, C, D) distinctly in each category based on a subjective ranking towards development advantage. For example, when we evaluate the cost-to-value for certain requirements elements, the grid is built as shown in Figure 4-4.

4.1 Characterization Framework

To characterize the need features, competition grids is built between two interrelated properties. This step is considered as the quantification process in which requirement feature/element *E* will be evaluated from two different perspectives. Entering many competitions creates a comprehensive evaluation that support the decision-making process.

Each competition-grid has set of class categories which supports different characteristics or properties. A matrix value is calculated and assigned each set two competed characteristics for one *E* requirement. Applying the same method for set of competition grids result in a definition of the matrix structure row/columns in which number of rows is defined by number of requirement elements E and number of columns is defined by number competition grids.

By listing all requirements entered for prioritization process and preparing all bounded properties that forms the definition of the competition grids, we evaluate subjectively each requirement element E against each gP which represent a competition grid and select the appropriate class in an independent of other requirements. we continue to repeat that evaluation in all identified competition grids gPn and perform similar thing with the rest of the requirement until we complete building the matrix. See (Figure 4-3).

As a result, this step concerns the building of a matrix is basically we build knowledgebase about requirements analysis results.

E1gp1	E1gp2	E1gp3	E1…	E1gpn
E2gp1	E2gp2	E2gp3	E2…	E2gpn
E3gp1	E3gp2	E3gp3	E3…	E3gpn
E4gp1	E4gp2	E4gp3	E4…	E4gpn
E5gp1	E5gp2	E5gp3	E5…	E5gpn
E6gp1	E6gp2	E6gp3	E6…	E6gpn
E7gp1	E7gp2	E7gp3	E7…	E7gpn
E8gp1	E8gp2	E8gp3	E8…	E8gpn
E9gp1	E9gp2	E9gp3	E9…	E9gpn

Figure 4-3: Competition matrix

4.1.1 Competition grid development

There are many properties can be considered for requirements characterization process. In our model, we identified twelve interrelated properties that can characterize the requirement systematically and with high quality model option.

- **Cost vs Value**

These two properties can be competed to distinguish the value added of developing the requirement considering the two perspectives. Cost can be calculated by staff hours (effort) or money spent compared with the revenue. Thus, in any form of cost/value calculation, the result must be in a form of normalized scale to map them easily in the strategic quadrants grid. Figure 4-4 shows that competition grid. There are different assigned matrix values which reflect the influence rate of the requirement (F). For example, if we have the requirements characterized as low cost with excellent ROI, then no doubt to give it a highest rate in the competition grid as “1” and “0.5” will be the second rate and the third will be “0.25” and so on

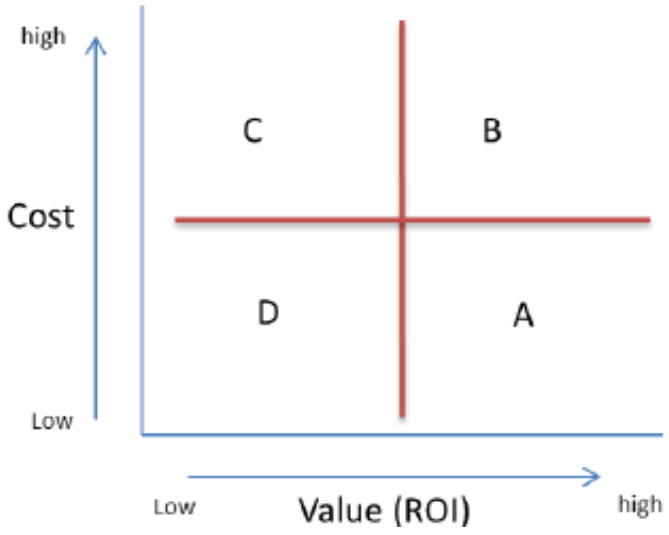


Figure 4-4: COST vs VALUE grid

- **Importance vs Urgency**

Stakeholders should identify which requirements are most important to the system users. Thus, Importance reveals the requirement implementation urgency which both of these two properties can be related to each other by definition. A competition guardant-grid will be setup as Fig 4-5.

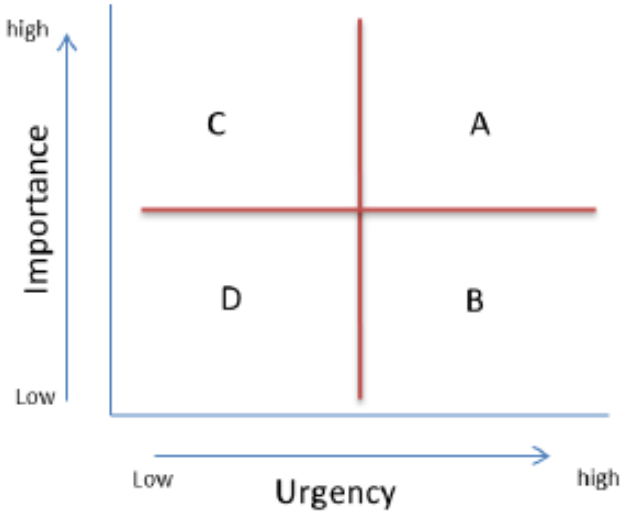


Figure 4-5: Importance vs Urgency

The highest matrices value will be for the requirement feature that is mapped under the category with highly importance rate with extreme urgency to be implemented in the next

releases and the same thing the second rate will be for the item that is less important but highly urgent.

- **Complexity vs Feasibility**

Software requirement complexity is one of the most important properties that should be examined in different aspects to forecast the implementation feasibilities. The complexity measurements involve requirement dependencies, coupling, volatility, effort, time, cost, scale, and variations. Therefore, whatever if it is calculated or subjectively evaluated; it must be normalized to the scale (0 to 1) in the competing quadrants grid which they are mapped as in Figure 4-6 to reflect the influence rate of the evaluated feature.

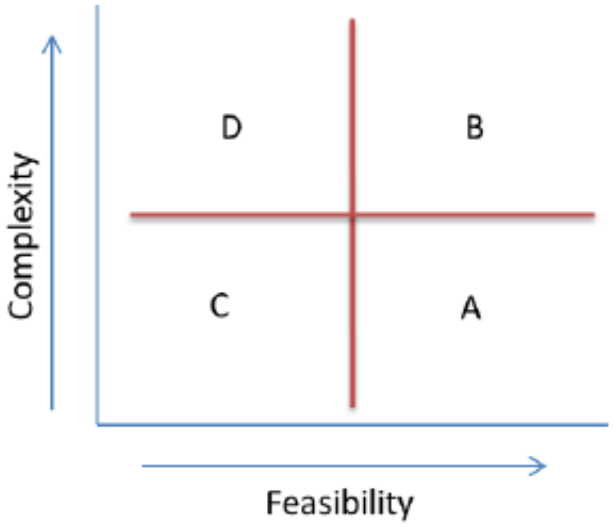


Figure 4-6: Complexity vs Feasibility

- **Risk vs Penalty**

Determining the level of risk of a requirement implementation is important which involve some performance risks, process risks, schedule risks, or introduce induced defects in the requirement development. On other hand, evaluating the severity of penalty when the selected requirement is not fulfilled is very important as well.

Therefore, two competing properties can be extracted and evaluated to come up with a clear vision about the challenges encountered during the implementation of the requirement. Figure 4-7 shows: the competition grid with assigned matrix value for these two associative-properties

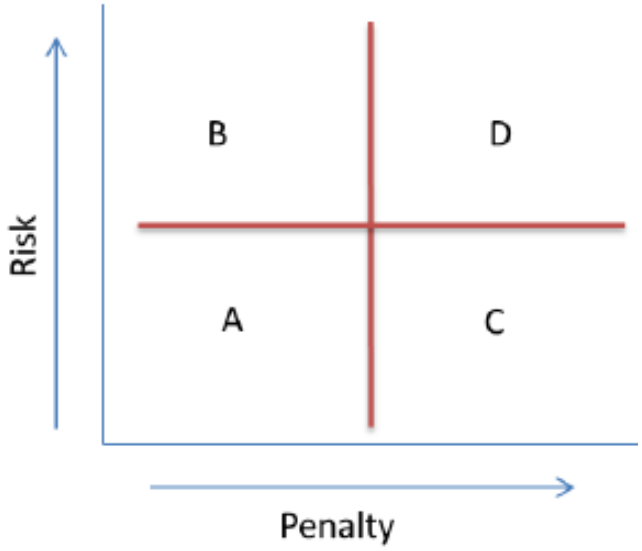


Figure 4-7: Risk vs Penalty

- **Volatility vs Stakeholders involvement**

Requirement volatility is property that is managed from different aspects: business requirements change, market changes, and users change. Requirements details become clearer during the development or design phases. Thus, the frequent requirements changes will affect the project plan and of course will have a negative impact on the costs. The involvement of stakeholders at the beginning is a key thing to minimize number of requirement volatility rate by negotiation and consolidation of the requirement conflicts. The two competing prosperities is plotted in grid with assigned matrix value. Figure 4-8 shows the relationship evaluation.

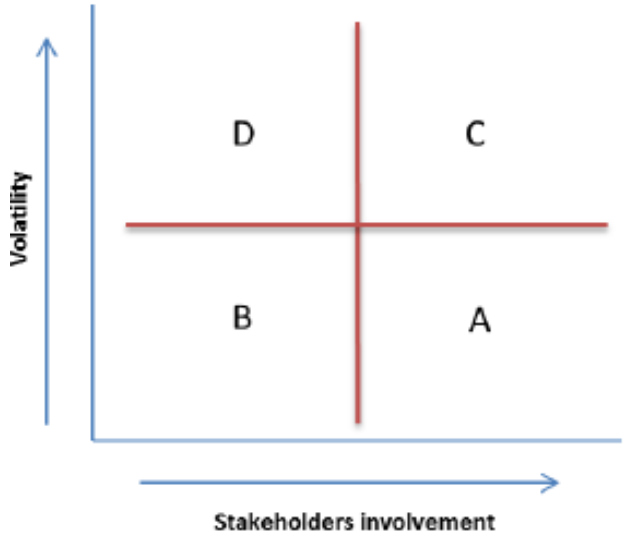


Figure 4-8: Volatility vs Stakeholders involvement

- **Competence vs Human Resources**

The skill set is a key factor that needs to be explored for the development of a requirement feature. These two competed properties will identify the level of expertise of a specific domain. For example, a requirement feature is documented as 3D drawing objects for the spare parts with several sizes for each car model, thus it requires high level degree of graphics programming. Therefore, as more resources with different skill set, more advanced requirement features can be implemented. A competition grid for the competence and human resources is shown in Figure 4-9

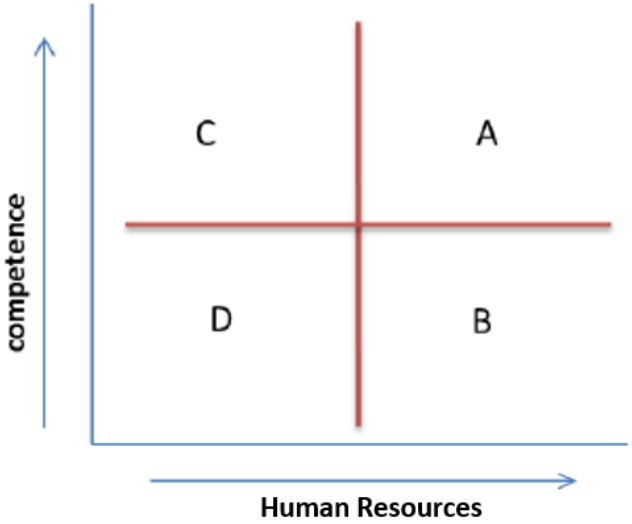


Figure 4-9: Competence vs Human resources

With an assumption of the competed grid properties that are listed *gpn* and the class types are defined and assigned in each *gqi*, a requirement element is evaluated in each of the listed *gpn* with an objective of placing the requirement elements in the right class category. We used a set of twenty requirement elements represented *E1, E2, E3, En* and applied the competition-based method on them as shown in Figure 4-9 below:

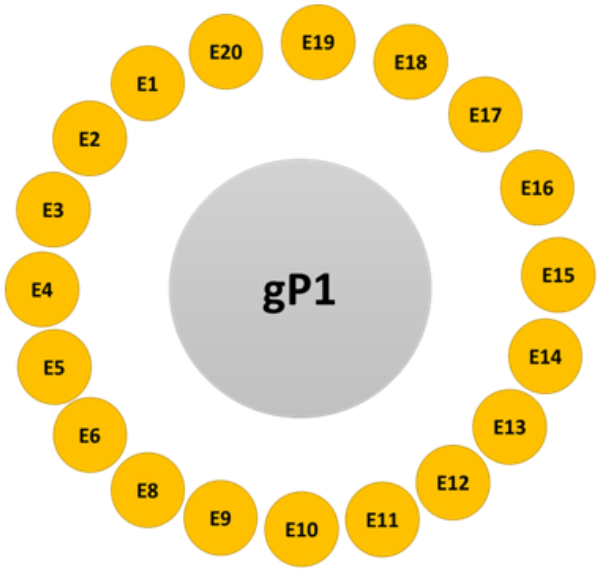


Figure 4-10: Competition graph

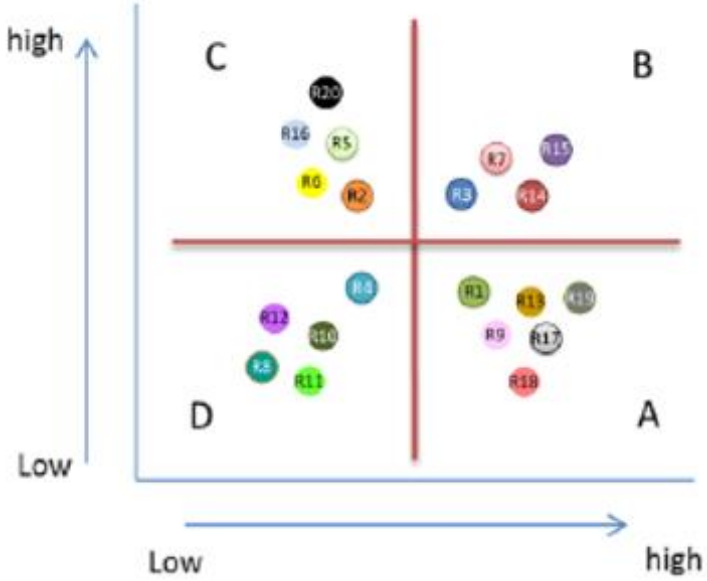


Figure 4-11: Competition grid example

Every grid property is a form of a competition model that requires each requirement element to be evaluated through its content.

$$Egp = \begin{bmatrix} E1 \\ E2 \\ E3 \\ E4 \\ E5 \\ E6 \\ E7 \\ E8 \\ E9 \end{bmatrix} \begin{bmatrix} gp1 & gp2 & gp3 & \dots & gpn \\ E1gp1 & E1gp2 & E1gp3 & E1\dots & E1gpn \\ E2gp1 & E2gp2 & E2gp3 & E2\dots & E2gpn \\ E3gp1 & E3gp2 & E3gp3 & E3\dots & E3gpn \\ E4gp1 & E4gp2 & E4gp3 & E4\dots & E4gpn \\ E5gp1 & E5gp2 & E5gp3 & E5\dots & E5gpn \\ E6gp1 & E6gp2 & E6gp3 & E6\dots & E6gpn \\ E7gp1 & E7gp2 & E7gp3 & E7\dots & E7gpn \\ E8gp1 & E8gp2 & E8gp3 & E8\dots & E8gpn \\ E9gp1 & E9gp2 & E9gp3 & E9\dots & E9gpn \end{bmatrix}$$

Figure 4-12: PageRank-based Matrix

With the new generated matrix *Egp* in Figure 4-10 as the evaluation results of each requirement elements.

4.1.2 Grid properties class counting

Apart of that characterized model, another generated matrix is presented to consider the weight factor for each element by counting the frequency number for each class type throughout the *Egpn* matrix. Since the importance of a requirement element weight measured based on its classes frequency, we can calculate the weights we assigned to the requirement element of the graph in a probabilistic way: getting a proportional value of each requirement element divided by the number of elements sharing the same values. This will ensure that weight factor measured proportionally to the level of and that is being calculated as follow:

$$Ew = \left(\frac{(E_{gP(i)})}{\sum_{t \in gPi} N(t)} \right)$$

Ew : requirement element weight
Egp: requirement element score
Nt: number of element sharing the same class type

	A	B	C	D
E1	A(EgPi)	B(EgPi)	C(EgPi)	D(EgPi)
E2	A(EgPi)	B(EgPi)	C(EgPi)	D(EgPi)
E3	A(EgPi)	B(EgPi)	C(EgPi)	D(EgPi)
E4	A(EgPi)	B(EgPi)	C(EgPi)	D(EgPi)
E5	A(EgPi)	B(EgPi)	C(EgPi)	D(EgPi)
E6	A(EgPi)	B(EgPi)	C(EgPi)	D(EgPi)
E7	A(EgPi)	B(EgPi)	C(EgPi)	D(EgPi)
E8	A(EgPi)	B(EgPi)	C(EgPi)	D(EgPi)
E9	A(EgPi)	B(EgPi)	C(EgPi)	D(EgPi)

Figure 4-13: Calculated weight factor matrix

4.1.3 Elements class-based clustering

In this step, we cluster the requirement elements in every performed grid model. We can model the process as a random walk on graphs as below.

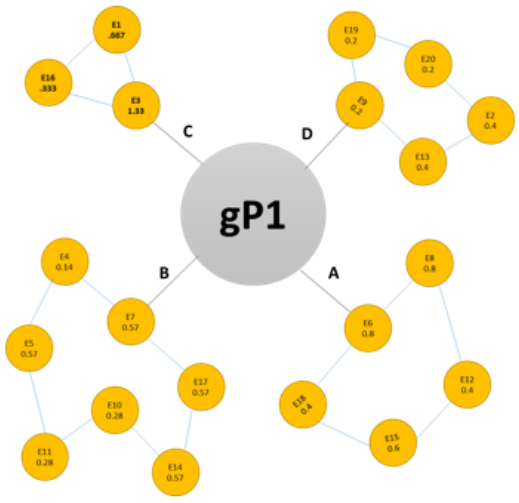
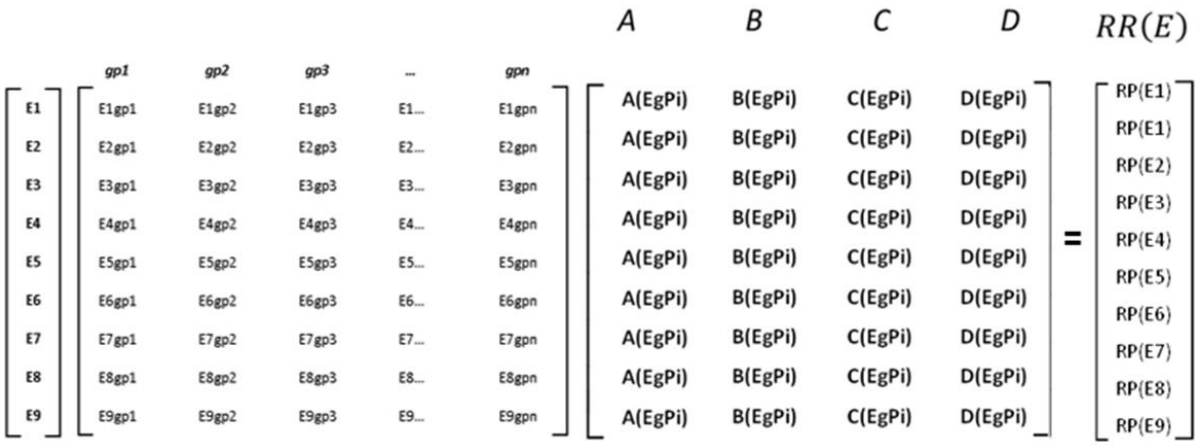


Figure 4-14: requirements clustering process graph

4.1.4 Elements Ranking

The calculated representation of each requirement element weight from the cluster graph *Egpn* gives initially each requirement element a certain position in the list. Then a summation of all Egpn gp properties for each element is performed and multiply by the actual of element property value. Below is a mathematical representation of the ranking process.



$$RR(E) = \sum_{i=1}^{gP_n} \left(\frac{(E_{gP(i)})}{\sum_{t \in gPi} N(t)} \right) \left(gP_{(i)} \right)$$

RR(E) : The rank of the Element E.
gP: grid property value.
$A(E_{gP(i)})$: Number of class for E requirements in $gP_{(i)}$ grid property.
N: Number of cluster node of type t.

\

4.2 Experiment

Worked with vendor A and selected 20 requirements/bug fixes randomly out of 634 requirements for one of their commercialized software from development releases plan (with 55% consistency issue (348 requirements) delayed).

. The dataset of this experiment study has been collected from the software vender with the priority list.

RF	Requirement	Description	Ranking	Serviwe delivery manger prioritization
R1	Seismic Compression	Velocity cubes and other non-zero mean (e.g., positive only values) data types should be compressed with a high (enough) value SQNR to maintain adequate signal quality.	2012.2.7	11
R2	SBG-Y 2D Toolbox	Upon clicking Define byte location, the Message log shows "Unexpected error occurred when resizing BinatryHdrGridview Exception: Index was out of range. Must be non-negative and less than the size of the collection. Parameter name: index". This message can be safely ignored. This message appears when working in a Windows environment where text size has been set at values other than 100% of normal size. WORKAROUND: Set Windows text size to 100% of normal size via Control Panel > Appearance and Personalization > Display	2013.2	25
R3	Multi-Z interpretation	Multi-Z interpretation is not yet supported within the Studio Environment. • Seismic operations run from the Operations TAB of the Multi-Z Interpretation cannot be undone. • Multi-Z interpretation projected point can become disconnected with the segment on the section (e.g., inline) while editing it on the crossing section (e.g., crossline). • The restrict mode is not available in Multi-Z interpretation	2012.2.7	4
R4	Mis-tie analysis	Realized output: When realizing mis-tie corrected seismic sections, the option "Scan for exact amplitude" is ignored when selected and the estimated min-max amplitude limits are used instead. WORKAROUND: Ensure the original sections have been scanned for exact amplitudes prior to running Mis-tie analysis, either individually or using the Survey Manager	2013	12
R5	Well top filter folder	Broken links between well tops interpreter attributes and the interpreter filter folder have been reported in 2013.2. These will now be repaired automatically when a project is opened using 2013.4. The root cause of this issue is still under investigation.	2012.2.7	3
R6	General	When opening old projects of Petroleum Systems Quick Look from 2009 and 2010, accumulations cannot be viewed with the intersection plane. Re-run the Charge process with the 2013 version.	2012.2.7	5
R7	Make generation properties	Rotated surfaces which were entered and resulted in an error message, when removed from the dialog still produce an error message.	2013.1	19
R8	Play Chance Mapping	Discrete surfaces are not supported by Play Chance Mapping, using a discrete surface as a property surface can lead to a Petrel crash. WORKAROUND: Use discrete surface attributes.	2013	13
R9	Imported PetroMod overlay: Gas content	There are differences in the definition of gas content in Petrel (mass/mass) and PetroMod (volume/mass) resulting in the wrong visualization of the overlay. Do not use this overlay.	2013.1	20
R10	Templates and units for "Temperature/Heat capacity" data	Petrel and PetroMod used different templates and units for "Temperature/Heat capacity" data. When PetroMod overlay data was imported using "Convert to simple grid" in the 2012.1 release, units were incorrectly converted. This is still an issue when you open overlays which were imported using 2012.1 versions, even with the 2012.2 or 2013.1 version installed. WORKAROUND: Re-import any overlays that were previously imported	2013.1	22
R11	Large 3D grids	An observation grid (reservoir grid) with more than one million cells will take approximately 10 minutes to compute during the Generate Fracture Driver process.	2013.2	23
R12	Fold related fractures	Models with fold-related fractures cannot give coherent results since the methodology is based on fault-related fractures.	2013.1	21
R13	Reference Project Tool	Some Petrel 2011 or 2012 projects might contain two Well section template folders with a warning symbol. Workaround: • Open the project in Petrel 2012 or 2013. On the Templates pane, right-click the template folder you want to delete, and then click Delete. Save and close the project. • When you reopen the project, there should only be one Well section templates folder.	2013	14
R14	Simulation Logs	When you upgrade a project from Petrel 2009 or 2010 to Petrel 2013, the simulation logs are upgraded as regular logs in the Well section window. Workaround: • Open the project in Petrel 2010. On the Input pane, clear the check boxes of the simulation logs, and update the template. Then, select the check boxes of the simulation logs on the Input pane, update the template, and save the project. • Open the project in Petrel 2009. On the Windows pane, expand a well section window, and delete the simulation logs. On the Input pane, select the check boxes of the simulation logs to re-add them again to the Well section window, and then save the project	2013	17
R15	Well plan display in an intersection window	If an intersection window is associated with a curved surface, the display result of certain plans might be incorrect	2013	15
R16	Converting a simple proposed well to an advanced plan	A simple proposed well is created with Well path design by manually digitizing the design points. When converting a simple proposed well to an advanced plan, the design points will be converted into Station points in the advanced plan. • If you click Go to surface or add any curve section to the top in Well path designer, for the Station points, only MD, INCL and AZIM values will remain unchanged during calculation, but X, Y, Z and TVD will be recalculated. • If the last point of the simple well (TD) is above MSL, the conversion will fail due to TVD and MD being referenced to MSL instead of to Kelly Bushing (KB). WORKAROUND: Edit settings for the proposed well by changing from simple well to standalone well with appropriate KB value	2013	16
R17	Associating a side track to a well	When assigning a branch to a main well during the sidetrack creation workflow, the trajectory of the main well (parent) and the branch (sidetrack) must be within 1m. Otherwise, the assignment will fail.	2012.2.7	6
R18	Well path creation with non S-3D profile	The system automatically converts any non S-section (profile) into an S-3D profile when you click on that section in a 3D window and in well path design Pick/Dragger mode. The conversion will display in the Well path designer spreadsheet as empty rows. Those rows will be filled up when modified with the dragger. WORKAROUND: Click the Undo button and return to the Well path designer.	2013.2	24
R19	Plan grid convergence values	In the 2012.1 release, if a project was set up with mixed units, the grid convergence for the plan might be calculated incorrectly. This issue was fixed in the 2012.2 release. However, if a project is saved with the 2012.1 version and is opened with any newer version, an incorrect grid convergence will occur. WORKAROUND: To correct this, you must trigger a plan recalculation by editing the plan (with Well path designer, dragger or Inspector) in a 3D window.	2012.2.7	7
R20	Transferring data over different index ranges	For some particular data sets, when one incoming log in the server contains multiple passes of data over different index ranges, if the deeper data range is downloaded before the shallower or earlier data range, the shallower or earlier data might be skipped during data transfer. For more details, see the online help.	2012.2.7	8

Figure 4-14: set of requirements features

We selected releases number to evaluate the consistency of our proposed model prioritized features. Since the release assignment is not our aim now, we prioritize the requirement according to releases assignments and for the equal releases assignments, we asked the software service delivery manger to perform a prioritization based on (Importance, added value, cost) for the listed twenty requirements.

To come up with the typical usage of our proposed model and reduce the single point of judgment and assigned a task to three developers to perform a subjective judgment for each requirement from different perspectives.

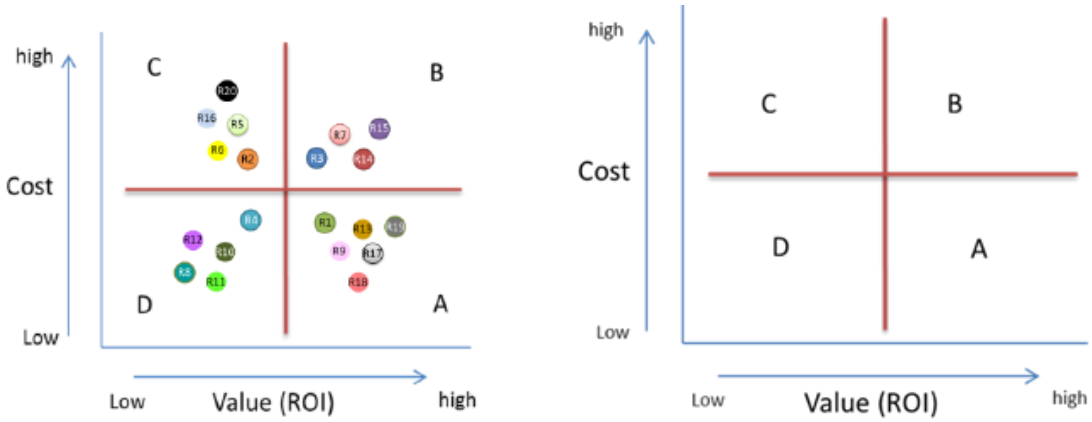
Step1: In this step, we collected the requirements set and select cost, value, feasibility, -complexity, and importance as aspects properties.

- Cost vs value
- Complexity vs feasibility
- Importance vs urgency

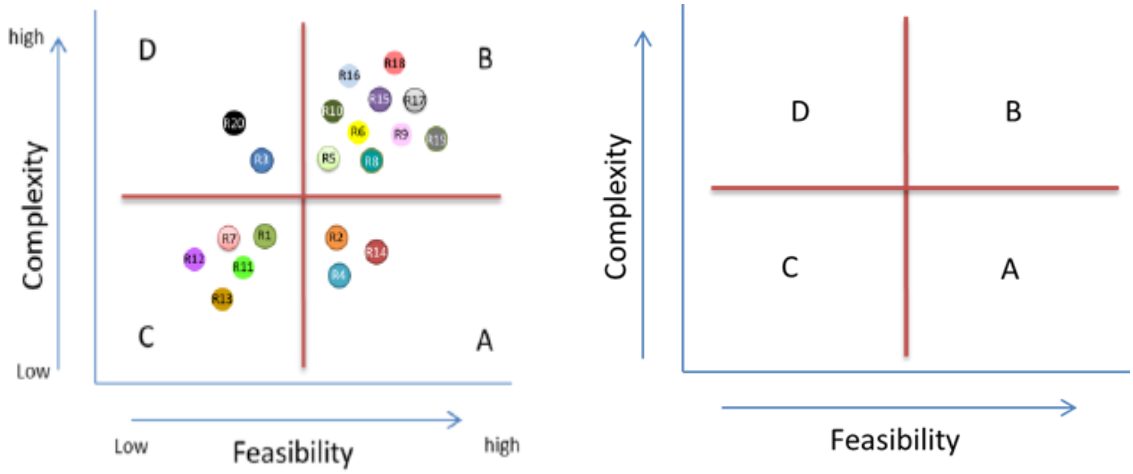
Step2: we build the competition grids for the above interrelated properties and we define the competition grids scores. Then, we performed the competition runs for all requirements,

Step3: we perform the clustering, sorting and prioritization.

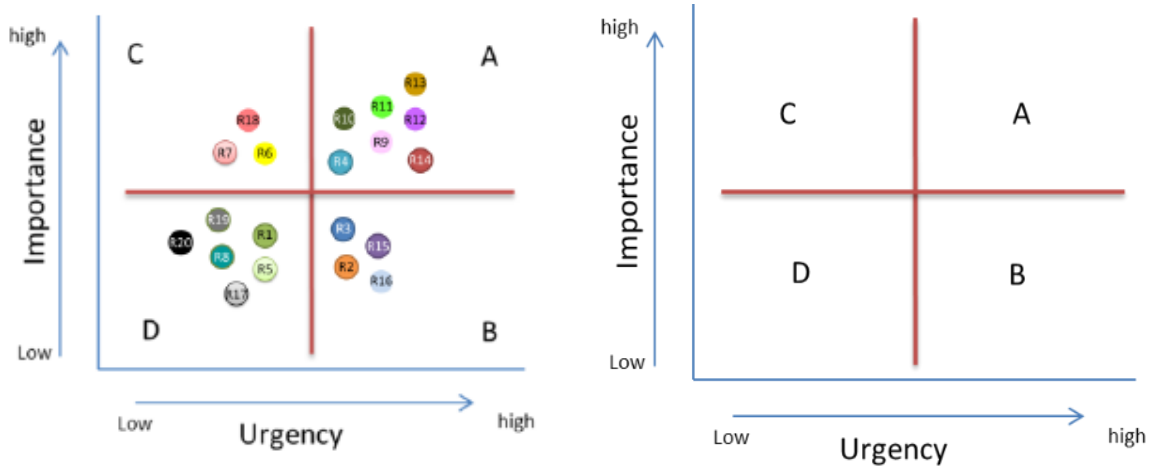
gP1



gP2


high
Complexity
Low
D
B
C
A
Low
Feasibility
high
Complexity
D
B
C
A
Feasibility

gP3


high
Importance
Low
C
A
D
B
Low
Urgency
high
high
Importance
Low
C
A
D
B
Low
Urgency
high

4.3 Experiment results

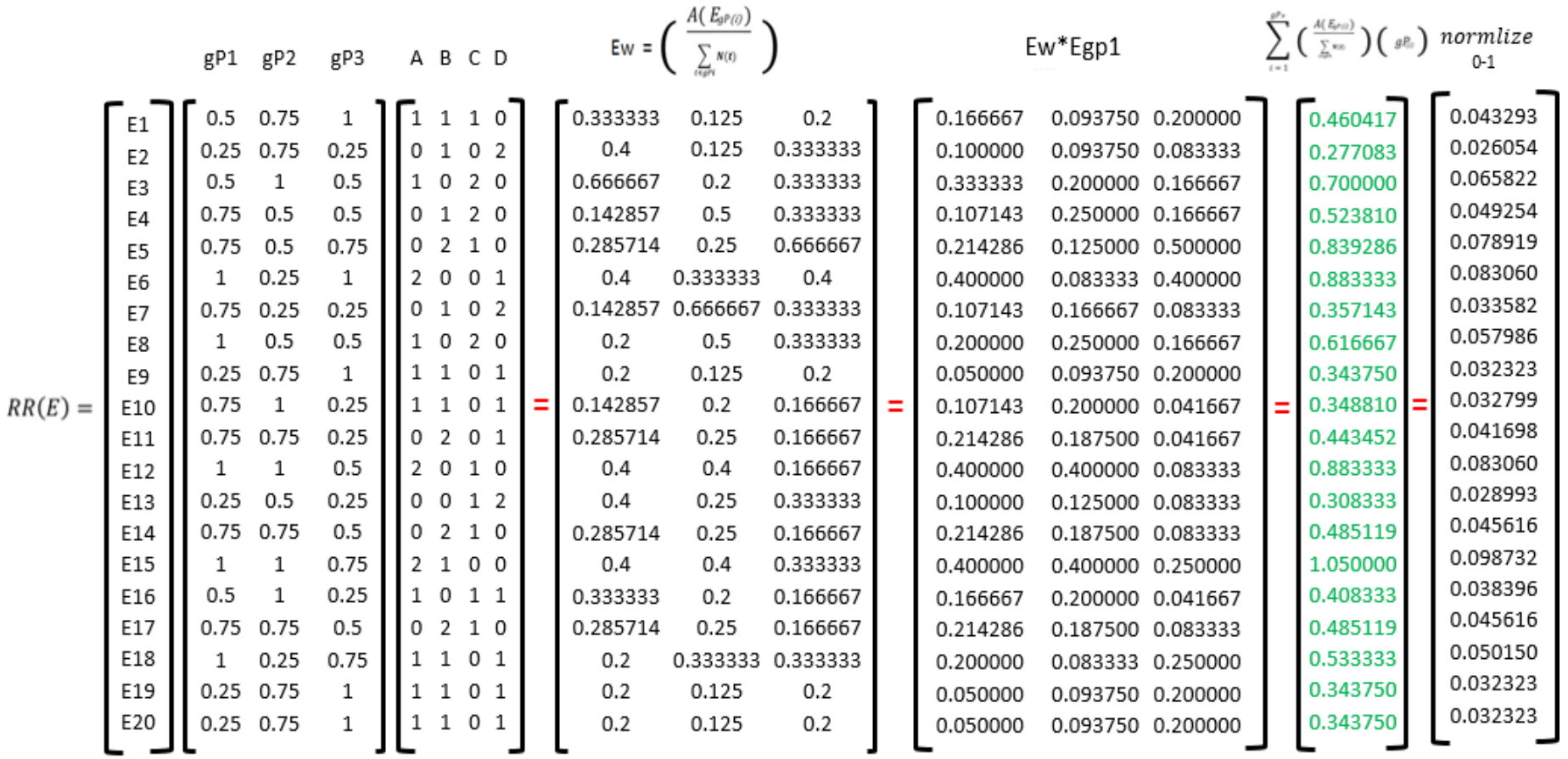


Table 4:2: competition grid and class count data (Experiment)

4.4 Results analysis

$RR(E) =$

Sorted element	
E2	0.026054
E13	0.028993
E9	0.032323
E19	0.032323
E20	0.032323
E10	0.032799
E7	0.033582
E16	0.038396
E11	0.041698
E1	0.043293
E14	0.045616
E17	0.045616
E4	0.049254
E18	0.05015
E8	0.057986
E3	0.065822
E5	0.078919
E6	0.08306
E12	0.08306
E15	0.098732

Element	Rank
E2	1
E13	2
E9	3
E19	4
E20	5
E10	6
E7	7
E16	8
E11	9
E1	10
E14	11
E17	12
E4	13
E18	14
E8	15
E3	16
E5	17
E6	18
E12	19
E15	20

The proposed framework Ranks

CONSISTENCY CHECK = 79%

	Planned release date	Actual release date	Reason
E1	2012.1	2012.1	
E2	2012.1	2012.1	
E3	2014.5	2015.3	dependency issue with E17
E4	2012.7	2014.1	required skillset
E5	2014.7	2014.7	
E6	2012.2	2012.2	
E7	2014.7	2014.7	
E8	2012.6	2012.6	
E9	2012.7	2012.7	
E10	2013.6	2013.6	
E11	2013.4	2013.4	
E12	2013.5	2013.5	
E13	2013.3	2013.3	
E14	2012.6	2012.6	
E15	2013.1	2013.1	
E16	2013.3	2013.3	
E17	2014.3	2016.5	dependency issue with E7
E18	2014.3	2014.3	
E19	2013.1	2013.1	
E20	2013.2	2015.3	cost-to-benefit

Release plan for 20 requirements

Table 4-3: Clustering, sorting, prioritizing process (left to right) (Experiment)

CHAPTER 5: EVALUATION FRAMEWORK

5.1 Introduction

In this section, we adopted a subjective evaluation method using a comparative analysis. We believe it is suitable method where a new proposed model that introduced as solution to address common issues related to requirement prioritization practices. There are huge number of prioritization methods/release planning models reported in the literature and validated through case studies [11]. The common applied evaluation method was designed to test the applicability in small size data set , this test is not enough to measure the characteristics of those methods/models from the quality point of view as it has been confirmed for a need to user a large and real data set [5][11][48l.

5.2 Comparison criteria

We conducted the evaluation using a comparative analysis and selected the following parameters as comparison criteria. The selection of those criteria was based on two reasons. First, identified as the common comparison criteria used in many literature studies and highlighted as challenges/issues. Second, some of these criteria were identified during out applied grounded theory research method as concepts and categories to be addressed in the proposed solution.

Below are the comparison criteria:

- Concept: It indicates how a method is applying the processes, following systematic and analysis approaches, rules, and recommendations employed by the prioritization method.
- Ease of Use: It indicates a characteristic of how a new user can easily use and apply the prioritization method.
- Size of data (Scalability): It measures the amount of requirement sets accommodated by the prioritization method a minimized human effort.
- Fuzziness: It refers to the uncertainty of human thought associated with the implantation of the prioritization method.
- Multi-criteria: It refers to the multiple factors considered during the prioritization method practice.
- Stakeholders-involvement: It refers to the multiple stakeholders involved with the application.
- Complexity: it measures the number of comparisons required to execute the prioritization method.

5.3 Comparative analysis

There are several requirements prioritization techniques acknowledged by researchers and applied widely in the industry [49]. We selected five RE techniques [2] [3] [4] [16] [28].

- Bubble Sort Technique introduced by Karlsson [3]. which uses a sorting algorithm that works by continually traversing through a list of elements to be sorted, comparing each pair of items and swapping them. If they are in the wrong order.
- Priority Assessment Method was introduced by Kunia [2] to assess the priority of requirements subjectively based on multiple aspects. Priority assessment method

uses a matrix to consider the relationship of multiple perspectives of stakeholders that utilize the concept of correlation to compute weighted priorities of requirements.

- Analytical Hierarchy Process (AHP) was introduced by Satty [16] to analyze and support complex decisions. AHP is a multi-criteria decision-making technique that uses a pair wise comparison matrix to compute the relative value of requirements with respect to one another.
- Fuzzy AHP (FAHP) [28] is a systematic decision-making method which includes both qualitative and quantitative technique to remove incompleteness, uncertainty and vague data.
- Requirements Triage [4].uses cluster based automated method expresses various concerns of stakeholders to multiple categories of requirements such as feature based, non-functional requirements and other cluster requirements.

The objective of this comparative analysis is to illustrate and evaluate the existing prioritization methods including our proposed model against several criteria defined in section 5.2.

Such as technique used, multi-criteria, multiple stakeholders and complexity analysis etc. as presented in Table 5-1.

PRIORITIZATION METHODS / EVALUATION ATTRIBUTES	BUBBLE SORT	AHP	FUZZY AHP	PRIORITY ASSESSMENT	REQUIREMENTS TRIAGE	CHARACTERIZATION FRAMEWORK
CONCEPT	End-to-end requirements Comparison	Pairwise comparison	Pairwise comparison	Relationship matrix	Clustering	Competition-based properties
EASE TO USE	Simple, easy to use & implement	Easy to implement	Difficult to implement in comparison to AHP	Difficult to implement	Complex to implement	Simple
SIZE OF DATA	Small	Medium	Medium	Medium	Large	Large
FUZZINESS	No	No	Yes	No	No	No
MULTI-CRITERIA	Not Supported	Not Supported	Not Supported	Supported	Supported	Supported
STAKEHOLDERS-INVOLVEMENT	Not Supported	Not Supported	Not Supported	Supported	Supported	Supported
COMPLEXITY	O (N^2)	O (N^2)	O (N^2)	O (N^3)	O (N^3)	O (N^2)*Number of competition tests

Table 5-1: Subjective Evaluation Sheet for RP Methods

5.4 Discussion

The objective of the evaluation is to make a comparison and evaluation of our proposed model among other selected RP methods. We will not argue that the results obtained in this evaluation can be generalized. Rather, we want to illustrate the prioritization practice to gain better understanding about some challenges that have been explored/addressed in some RP methods and have not been addressed at all. In addition, selecting a prioritization method must be based on quality objectives

As we know, that RP practices aim to select the 'right' requirements from the set of candidate requirements so that all the different key aspects such as technical constraints, business constraints, and client preferences of all stakeholders are satisfied. In Table 5-1, it shows how the RP methods are performing against each attribute. Each analyzed are quite enough and has it is own cases of practice. Looking at the concept attribute and scan the results, it gives an indication about the fitting of RP methods under two categories. First category, the concept is about identify which requirement element is outperform the other from one or different point of views. Other category is that some RP practices uses multi factors to build a matrix to calculate the rank of a requirement.

In relation to with the issues associated with requirement quality subject to the prioritization practice, Table 5-1, the concept them of most of them was the pair-wise comparisons towards serving one goal of enforcement. Thus, we don't see there is a need to follow it in our proposed model since we know the requirement characterization process will reveal the importance of each requirement elements when it considers different factors.

Another important criterion is *Ease to use which* reveals a very important aspect of a method. The judgment of this criterion was based on how much efforts are required to prepare the requirement set and use the method. The implication in this criterion is to tell us how simple it is to apply it. In

Table 5-1 among all the evaluated method, we found the characterization framework is among the simple ones. It requires only two more steps that are essential and simple to be adopted for the practical use of the framework.

The size of the data is one important criterion aspects that determine the ability of a method to handle a growing number of requirements. The challenge here is as the size of software intensive systems continues to increase, which can have produced a bulky set of requirements. The number of requirements may range from very.

Multi-criteria attribute is another important criterion, which defines the aspects or factors that are considered during the prioritization practice. In the characterization framework, we considered the adaptation of several aspects in building the competition grids. Thus, quantification a requirement feature is performed within the adaptation of different competed properties types.

Additionally, another important criterion that measures the stakeholders' involvement during the decision-making process. The consideration of that with the frequent requirement changing practice is critical to capture the all details of requirement feature specification.

Finally, it was observed that studying the use of the characterization framework method in the industry is the key important thing. This will serve as an additional source of validation for the framework, and may also reveal other factors that need to be addressed. Furthermore, we might to empirically validate the method using a large data set from the real-world industry.

CHAPTER 6: CONCLUSION

The aim of this research work was to empirically develop a multi-criteria requirement characterization framework to address the inconsistency issue between actual software deliverables and the software release plan. The research work was conducted in three different forms of studies including interviews with subject matter experts (participants), document analysis for some previous software development projects, and observation of an on-going complex software development project. The method utilized in this research was the grounded theory (GT) approach to explore and investigate aspects of the requirement prioritization practice from three data sources.

In relation to the GT applied method, some valid research questions were articulated based on the aim of the study, to identify and discover the root causes and implications of the inconsistency issue and its related concerns. The initial focus of this research study attempted to understand how software companies adopt their prioritization practices, build their own requirements selection criteria process, identify missing characteristics of the RP practices, understand causes of project failures of high frequency requirements updates, and understand to which level of abstraction the RP practice can be maintained. The essence of these focus areas was useful in exploring RP practice shortcomings for possible improvement or enhancement. This was done via analysis of the collected data from interviews, document analysis, and literature reviews. Apart from document analysis and

literature reviews, the interviews conducted were limited to one open-ended question to enrich the discussion and explore thoughts and clarifications from the participant. With this approach, it was possible to trigger other minor questions and seek more reliable answers that provided an opportunity to restrict biases when answering inapplicable questions to the participants.

In this research, we followed established grounded theory workflow processes to systematically reach conclusions while maintaining the area of focus and keeping extracted concepts and categories retained. GT workflow process began with the data sampling process from data sources, which were used as input for the data analysis (coding) process. This analysis process immediately revealed several concerns related to the selection of the best RP practice, its applied selection factor(s), and its limitations, which introduced other issues for detailed investigation. Iteratively, the process was repeated to collect more data about each of the introduced issues, analyze them, and extract the new concepts and categories. This was repeated until saturation level after which the discovered RP concepts and categories were formally compiled. Findings were mapped from concerns/challenges to concepts and their related issues were mapped to categories. Because of applying GT methodology, it was possible to identify concepts/concerns for the inconsistency issue and its implications in terms of multi-criteria, scalability, practicality, accuracy, interdependency, and ease of use as categories for evaluation attributes. In addition, the reasoning behind RP practice selection preferences performed by software companies and their business drivers were identified, which highlighted the domination of one-dimensional aspects in most RP practices.

Besides these GT-based research findings, a new concept was introduced for requirement characterization activity including an empirically developed competition-based framework to facilitate the RP practice and address the inconsistency issue during the requirement engineering process phase for a commercial system development in the Oil & Gas industry. Key structural

characteristics of the proposed framework were described, based on the highlighted theoretical findings and practical adoption aspects applied for release planning purposes.

Other research objectives were achieved by analyzing the influence of each RP factor and identifying interrelated RP factors towards the success of the release planning. Also investigated were the relationships between various factors to evaluate the size of contribution to the requirement prioritization practice.

In addition, a simulation experiment was conducted using small real requirements set and applying the proposed characterization framework. The data set used in this experiment was for a successful system development project that achieved 45% of its consistency degree, between the developed requirements (deliverables) and the initial release plan. The results showed an exact match that revealed a significant response in stabilizing the consistency attribute and achieving a high degree of optimality in considering a multi-criteria decision-making process. We believe this proposed framework is promising and can be useful for adoption by software development companies and in-house software development organizations.

Moreover, a comparative analysis was conducted to subjectively test the performance of our proposed framework from different perspectives, among other requirements prioritization techniques acknowledged by researchers and applied widely in the industry. We selected a set of comparison criteria based on commonly used evaluation criteria for requirements prioritization techniques in literature studies including (*concept, ease of use, and size of data, fuzziness, multi-criteria, stakeholders-involvement, and complexity*). This analysis is not intended to argue the generalization of the proposed framework, but to show how the framework performs when it is evaluated with other techniques around those evaluation criteria. The result gives a better

understanding regarding concerns that have been explored and addressed. It also introduced an area of improvement related to time complexity. The analysis gives an indication about considering other quality attributes that ultimately contribute to the success of any requirements prioritization practice. Finally, it becomes clear that the requirement prioritization practice is an essential task during the requirement engineering process phase. RP practice must be performed as a major step within the requirements analysis workflow. Implementing the prioritization practice based on multi-aspects is a crucial factor towards the success of the decision-making process.

6.1 LIMITATIONS

The initial analysis of the characterization framework revealed complexity in terms of the number of comparisons needed to characterize a set of requirements. It depends on the requirement analysis for dependency factor and considers a requirement feature as a low-level abstraction thus; it does not handle dependencies between requirements. We also adopted a kind of subjective evaluation model that can be debatable. In this case we need to conduct a case study using a large-size data set to confirm the proposed framework among other RP method and release planning models.

The subjective evaluation was not intended to argue the adoption of this framework, but to show how the framework performs among other RP techniques criteria. Hence, a case study using a large-size requirements set is required to evaluate the adoption and practical use of this framework.

Another point that might be arguable is the level of the framework involvement to solve a conflict in prioritizing the requirements when there are many participants in RP practice within a specific company. In the proposed framework, we assumed a company/stakeholder should come up with only one conflict-free response for the priority list of the requested requirements. It doesn't go further to the level of each company (client) and solve the conflict within a company itself.

6.2 FUTURE WORK

This study focuses on proposing a framework for requirement characterization from a general perspective. Since we discussed what has been highlighted from GT study as concerns, concepts, or core categories, some related categories may have been left out. These categories may have no direct effect to the RP practice. However, more analysis is needed and some research work can be defined to;

1. Conduct a case study using a real-world large data set for the practical use of the framework purpose and validate the proposed model from the following quality attributes point of view;
    - Scalability: is a quality attribute that defines the ability of the selection method to handle a large set of requirement features that are required to be processed using the proposed framework. Most of the proposed techniques struggle to satisfy this attribute.
    - Error proneness: is a quality attribute that measures the negative impact of rule settings in the prioritization technique. This has led to the generation of unreliable prioritization results because results do not reflect the true ranking of requirements from a stakeholder's point of view or assessment after the ranking process.
    - Computational complexity: is a quality attribute that defines how much time/effort is required to determine their ease of use and accuracy.
    - Rank updates: is a quality attribute that defines the ability of a technique to iteratively and automatically update a requirement list's volatility "anytime" a requirement is included or excluded from the list, "anytime" prioritization.

- Reliability: is a quality attribute that defines exclusively the requirements dependencies considerations towards the success of each requirement feature implementation.
- Comprehensiveness: is a quality attribute that measures the satisfaction of different aspects in the prioritization method and implementation of the non-functional requirement.

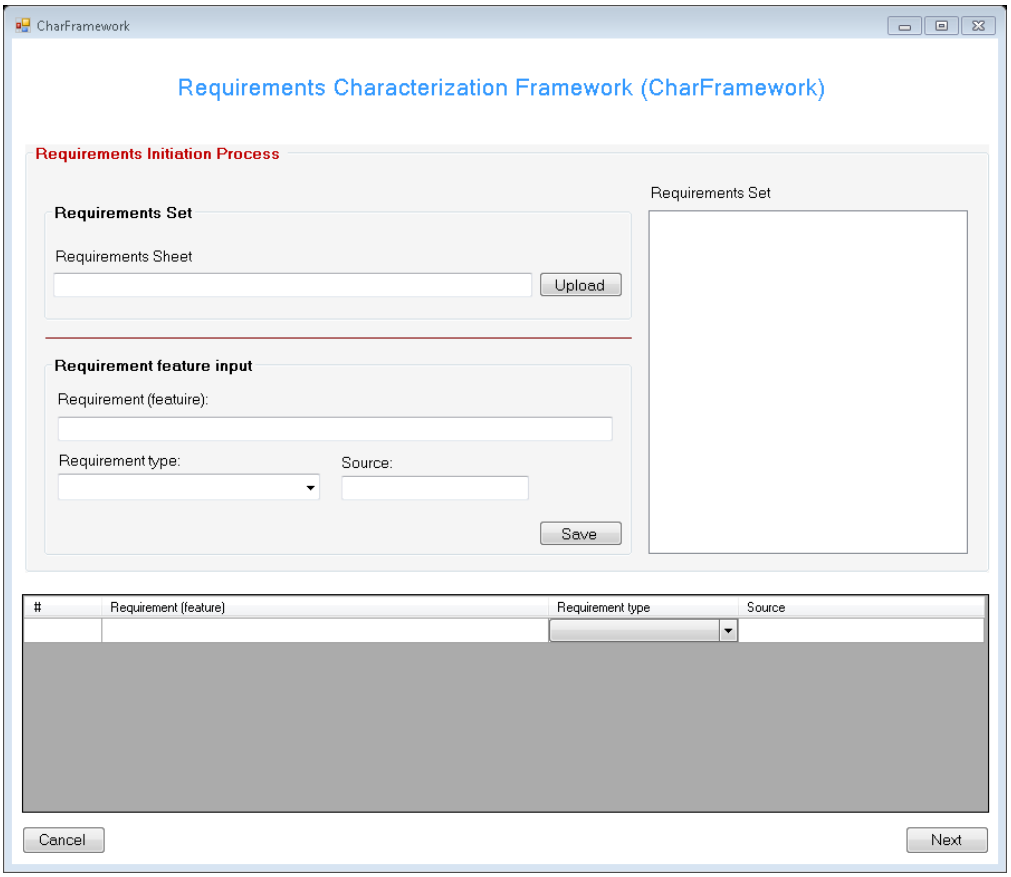
APPENDIX

A. Software Prototype Tool for The Characterization Freamwork

This appendix presents the tool that we developed as a prototype for the characterization framework. This tool is presented in three sections according to the number of UIs.

- First UI window

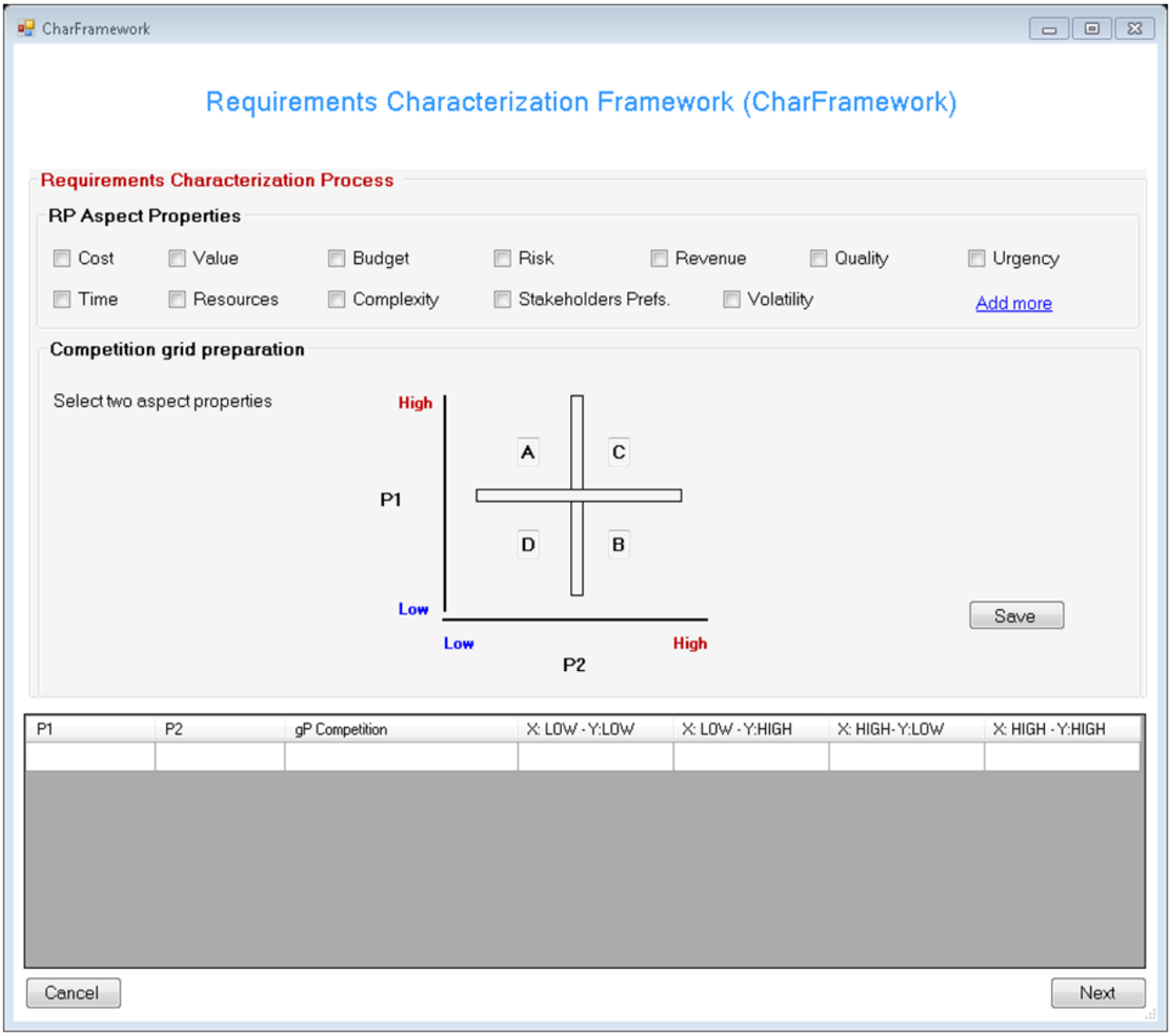
This is the user interface for the requirement initiation process of characterization framework uploading the requirements set as batch from a file with three input parameters associated with every requirement feature. Requirement types determine (functional, non-functional, business requirements)



- Second UI screen

It is the UI implementation of the second process in the framework that focuses on selecting the properties and defines the competition grids and score classes.

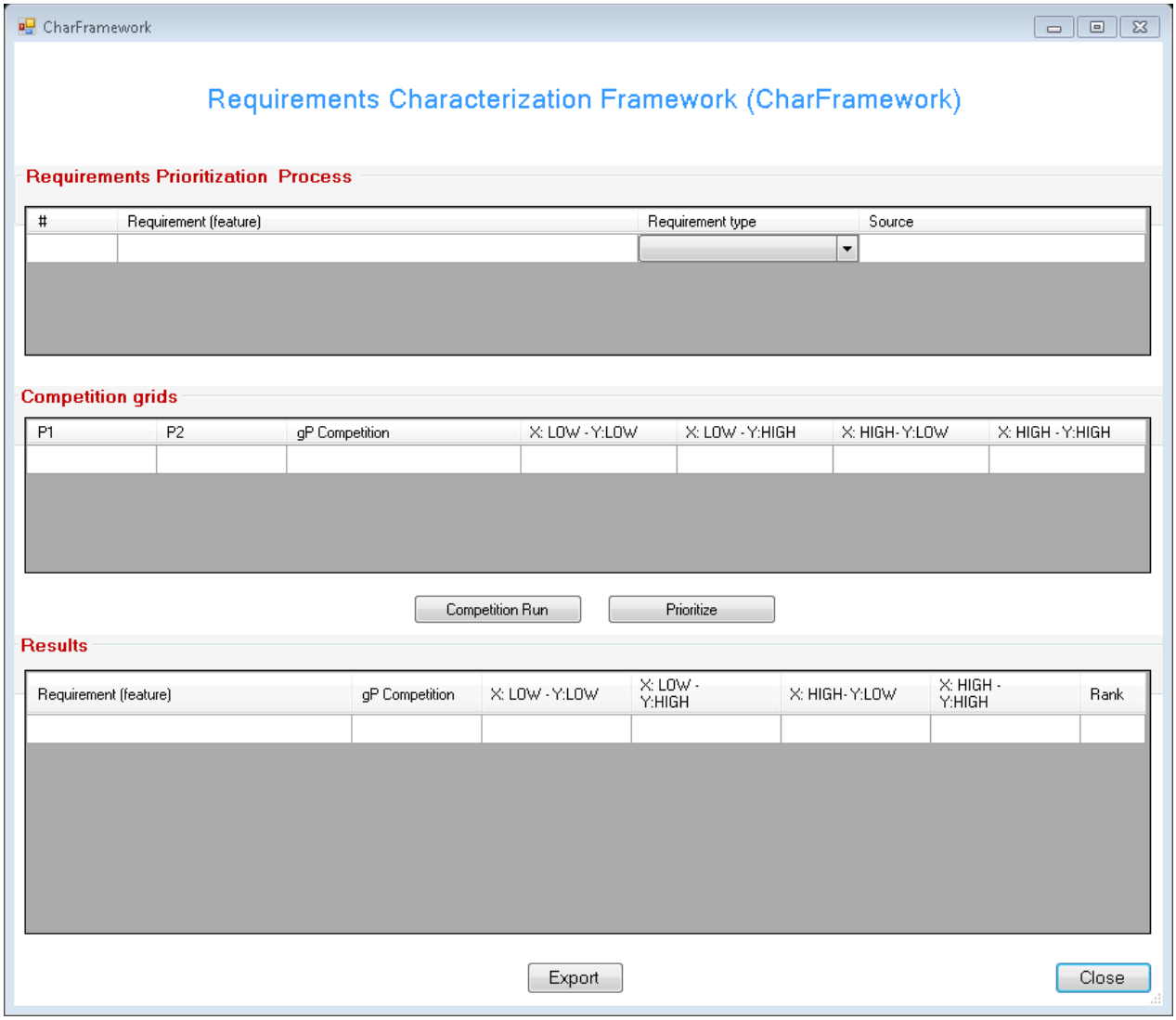
- User can add another property (factors) and incorporate into the competition grids.
- The class assignment should be based on the contribution direction.



- Output UI

  This the final process of the framework workflow in which the competition is performed, clustering is developed and prioritization process is carried out.

  - It produces the rank of requirement set.



B. Descriptions and limitations of existing prioritization techniques

#	Name of techniques	Description	Limitations/source
1	Analytic hierarchy process	Pair-wise comparison matrix to calculate the relative importance of each requirements	Time consuming, Not scalable
2	Attribute goal-oriented requirement analysis	Attributing preference values to requirements computed in a decision matrix form and represented in a goal graph	complex
3	Benefit and cost prediction	Based on the value of each requirement to the organization or customer	Not cater for requirements growth
4	Binary Search Tree	Ranks requirements them in a hierarchical order (parent-child relationship)	Only for simple requirements ranking
5	Binary-tree	Search for requirements in nodes and compare them to determine relative	Complex, scale issue
6	Binary priority list	Rank requirements based on their real benefits to the application domain	
7	Bubble sort	- Prepare and arrange requirements in a vector. - Execute of the requirements comparisons.	Not scalable
8	Case based ranking	Machine learning based approach to reduce the amount of information required from stakeholders	Not scalable; inability to support coordination among different stakeholders
9	Correlation-based priority assessment framework (CBPA)	Prioritize requirements by incorporating inter-perspective relationships (correlations) of requirements using relationship matrix	negative correlations issue
10	Cost-value ranking	Prioritize requirements based on their perceived value and implementation cost	Time consuming and Not scalable
11	Cognitive driven	combines the use of AHP, self-organizing maps requirements prioritization (SOM) and Cognitive Psychology methods to build a Conceptual Framework	Lacks aggregating and globalizing preference weights.

12	Dot voting	Uses sticky dots to rank requirements. The higher the numbers of sticky dots, the most valued a requirement will be	
13	Eclipse process framework	rank requirements with a weighting scale	
14	EVOLVE	uses genetic algorithm in a number of iterations to maximize the weighted benefit over all the different stakeholders	Computational complexity and needed to be tested in a more complex industrial setting
15	Fuzzy AHP	uses triangular fuzzy numbers and weights for ranking requirements	Not scalable, lack of requirements interdependencies
16			
17	Hierarchy AHP	Requirements are prioritized in hierarchical order based on the accumulated scores of each requirement across relevant stakeholders	judgmental errors
18	Interactive requirements prioritization	utilizes a weighting scale to prioritize requirements	No experiments
19	Kano model	comparison of customer satisfaction with technical Excellence	
20	Lanchester theory	Business objective and market share. similar to quality functional deployment technique	No definition of relative values
21	Mathematical programming techniques	Use machine learning algorithms	
22	Minimal spanning tree	utilize a weighting scale to rank requirements in a hierarchical form	judgmental errors due to inconsistencies issue
23	MosCow	‘‘MUST have’’: Requirements are not negotiable; “SHOULD have’’: Features that would be nice to have if at all possible ‘‘COULD have’’ Features that would be nice to have	

		‘‘WON’T have’’ These requirements are not important,	
24	Multi voting system	stakeholders vote for requirements in line with their real importance of each requirement	
25	Multi-criteria Preference Analysis Requirements Negotiation (MPARN)	Guides stakeholders to make negotiated agreements using multi-criteria preference analysis techniques.	inconsistencies issue
26	Multi-objective next release problem	utilizes the main objectives behind the development of software	Only group the requirements, does not produce an order for requirement.
27	Numerical assignment	grouping requirements into different categories, viz. high, medium, and low	No practical
28	Pair wise analysis	comparing requirements in pairs until the top requirements emerge at the top of the stack	complicated and not reliable
29	Ping Pong Balls	ping balls represent each requirement are given to stakeholders to cast their lot in order of requirements priority	
30	Planning game	Clients categorize requirements into three classes: i.e.; Essential, conditional and optional.	Not scalable
31	Priority groups	formation of priority groups, requirements are classified based on their importance by stakeholders	inconsistencies issue
32	Quality Functional Deployment (QFD)	Matrices to chronologically represent client’s expectations and how these expectations are to be met by the developers	Good small systems, inconsistencies issue , not scalable
33	Ranking	This technique implores numbers to rank requirements in ascending order starting from 1 to n	Not scalable, suitable for single stakeholder
34	Rank based on product definition	accounts for three important perspectives on product definition: the business, users, and technology	

35	Relative weighting	rank requirements using a weighting scale	
36	Requirements triage	Determines requirements priorities by educating stakeholders to understand each requirement before ranking commences.	Prone to errors
37	Requirement uncertainty prioritization approach (RUPA)	Aggregating weights using internal evidential reasoning (IER) algorithm	Not scalable, good for small scale project
38	Round-the-Group Prioritization	Requirements are displayed on cards and positioned in a haphazard manner where stakeholders are requested to reposition the cards according to their order of preferences	Not scalable
39	Simple multi-criteria rating technique by swing (SMARTS)	weighs requirements based on some defined criteria	inconsistencies amongst ranking values, Not scalable
40	Software engineering risk understanding and management (SERUM)	Uses estimations for cost, benefit, development risk, and operational risk reduction	does not handle dependencies
41	$100 Allocation (Cumulative Voting)	Stakeholders are given a $100 note to expense on the elicited requirements.	Not scalable
42	Technique for ordering from similarity to ideal solution (TOPSIS)	Requirements are valued based on a weighting scale	No Rank Update
43	Theme screening/scoring	scoring requirements using a pre-defined criterion for the scoring process	
44	Top ten requirements	Stakeholders are asked to choose their top-ten requirements from the pool of requirements	Uncertainty is high since weights are not utilized in the ranking process

45	Value based requirements prioritization	Extracts individual utility functions from each stakeholder	Inability to hierarchically organize goals or requirements
46	Value oriented prioritization	business values and prioritized based on the stakeholder ratings	Disregards requirement dependencies, Not scalable
47	Value based intelligent requirement prioritization	use of stakeholder's, experts and automated fuzzy logic	Nott clear-cut step to classify the ranked requirements
48	Weighted critical analysis	Criteria are established and weights are allotted	Disregards requirement dependencies
49	Wiegers' matrix approach	A semi-quantitative analytical that uses a simple spreadsheet to estimate requirements' priorities	Not secure- manipulated by stakeholders
50	Win-Win	Express goals as win conditions	inconsistencies

Table B: descriptions and limitations of existing prioritization techniques

C. Requirements Selection Factors for release planning models

Software Release Planning Model	Selection Factors																	
	Stake Holder Preference	Cost	Value	Effort	Time	Requirement/Feature Dependency	Resource	Risk	Budget	Revenue	System Constraint Product Feature	Deadlines	Business/Market Values	Quality	None Functional Requirement	Urgency	Fuzzy Constraints	Development Team
CVA	*	*	*															
IFM		*		*	*													
EVOLVE	*			*		*												
EVOLVE+	*			*		*	*	*										
EVOLVE*	*			*		*	*		*									
F-EVOLVE		*			*	*	*			*								
EVOLVE EXT	*				*	*												
S-EVOLVE	*					*	*				*							
NRP	*	*				*			*									
AHPSRP	*			*	*				*									
MORP	*	*				*	*	*				*						
MONRP	*						*											
BORPES	*					*							*					
AEQWW	*			*	*									*				
AMRSQE				*		*			*				*					
REPSIM				*			*											
QUPER		*													*			
AMFFRP						*	*					*						*
RDMXP-RP		*	*	*		*							*					
PARSEQ	*	*	*			*	*											
MMASDRW	*					*	*									*		
RPUFEC				*			*										*	
QIP				*		*	*											
AOTRP						*	*			*								
FMDRCP																		*
CDVBRPA					*	*										*		
FOMRP				*		*	*											
DIALOGUE APPROACH IN RP				*		*												
RPFT	*			*	*						*							
RP&CP	*					*	*								*			

Table C: Requirements Selection Factors for release planning models

D. Quality attributes for the prioritization practice

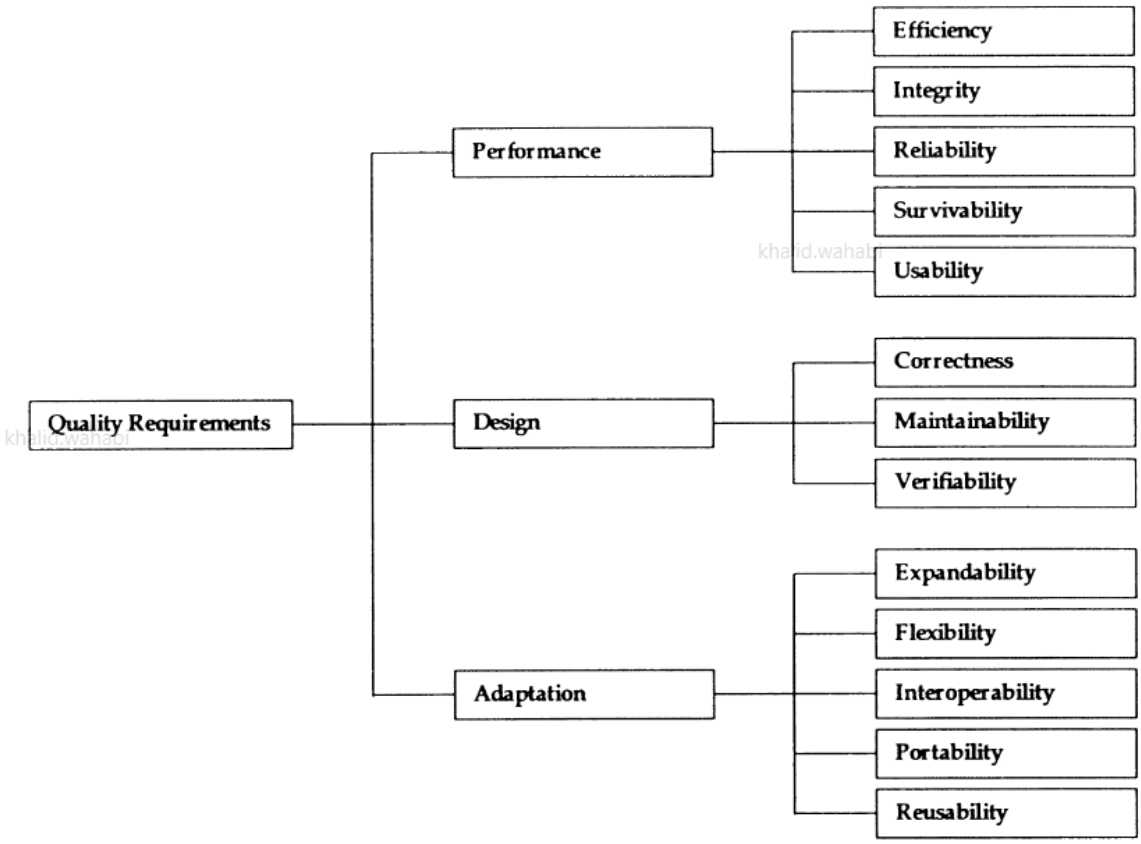


Table: D. Quality attributes for the prioritization practice

E. Experiment Data Set

A release planning data set for a commercialized geoscience software with 5K+ requirement created in a table format.

Requirement ID	**Requirement description**	**Analysis**	**Release plan**	**Actual release date**	**Contact**	**..**	**…**	**..**	

Table E: Experiment Data Set

Sketches, drawing, and memos developed while conducting GT study is also is included and image files.

For reference, the data set is attached with the softcopy of this document.

References

[1] A. Perini, A. Susi and P. Avesani, A Machine Learning Approach to software requirements prioritization, Software Engineering, IEEE Transactions on 2013

[2] P. Berander, P. Jönsson, Hierarchical Cumulative Voting (HCV) - Prioritization of Requirements in Hierarchies', International Journal of Software engineering and Knowledge Engineering (IJSEKE) on 2006

[3] J. Karlsson, C. Wohlin and B. Regnell , An Evaluation of Methods for Prioritizing Software Requirements Information and Software Technology, 2005

[4] J. Karlsson , K. Ryan, A Cost-Value Approach for Prioritizing Requirements, IEEE Software Journal 1997

[5] Z. Racheva, M.Daneva, K. Sikkel , A. Herrmann,R. Wieringa; Do We Know Enough about Requirements Prioritization in Agile Projects: Insights from a Case Study;Requirements Engineering Conference (RE), 2010

[6] C. Larman and V. R. Basili, "Iterative and incremental developments: a brief history," Computer Journal, 2003

[7] M. Svahnberg, G. Tony, R. Feldt, R. Torkar, S. B. Saleem, and M. U. Shafique, "A systematic review on strategic release planning models," IST, 2010

[8] G. Ruhe Software Release Planning, Advances in Software Engineering and Knowledge Engineering, 2008

[9] S. Amir, D. Rodina Ahmad, Software release planning challenges in software development: An empirical African Journal of Business Management 2012

[10] L. Sullivan, Quality function deployment: A system to assure that customer needs derive the product design and production process, Quality Progress, pp. 39-50, 1986.

[11] B. Boehm, R. Ross, Theory-W Software Project Management: Principles and Examples." IEEE Transactions on Software Engineering, 1989

[12] K. J. Stol, P. Ralph and B. Fitzgerald, "Grounded Theory in Software Engineering Research: A Critical Review and Guidelines," 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE), Austin, TX, USA, 2016, pp. 120-131

[13] J. W. Bracket; Software Requirements; Software Engineering Institute, Carnegie Mellon University, 1990.

[14] P. Berander, K.A. Khan, L. Lehtola, Towards a research framework on requirements prioritization, Proceedings of Sixth Conference on Software Engineering Research and Practice in Sweden , 2006.

[15] Petrel E&P : www.software.slb.com/products/platform/Pages/petrel.aspx

[16] T. L. Saaty The Analytic Hierarchy Process,McGraw-Hill, New York;(1980)

[17] J. Karlsson, S. Olsson, and K. Ryan. Improved practical support for large-scale requirements prioritizing. Requirements Engineering, 2(1):51{60, 1997.

[18] L. Lehtola , M. Kauppinen, Empirical Evaluation of Two Requirements Prioritization Methods in Product Development Projects, Springer-Verlag, 2004.

[19] A. Herrmann, M. Daneva, Requirements Prioritization Based on Benefit and Cost Prediction: 16th IEEE International quirements Engineering conference, 2008.

[20] H. Kaiya, H. Horai M. Saeki AGORA: Attributed Goal-Oriented Requirements Analysis Method, IEEE International Conference on Requirements Engineering, 2002

[21] A. Davis, The Art of Requirements Triage; IEEE Computer Juornal 2003

[22] K. E. Wiegers, Software Requirements, 2nd Redmond, WA: Microsoft Press, 2003

[23] F. Moisiadis, Prioritizing Scenario Evolution; International Conference on Requirements Engineering 2000

[24] M.I. Babar; M. Ramzan; S. A. Ghayyur, Challenges and future trends in software requirements prioritization, Computer Networks and Information Technology, 2011 international Conference 2011

[25] G. Kaur, S. Bawa ; A Survey of Requirement Prioritization Methods ; International Journal of Engineering Research & Technology; 2013

[26] M. Harman, P. McMinn, J. T. Souza,S. Yoo ; Search based software engineering: techniques, taxonomy, tutorial,Empirical Software Engineering and Verification. Springer, 2012

[27] A. J. Bagnall,V. J. Rayward-Smith ,L. Whittley.The next release problem ,2001

[28] Vibha Gaur, Anuja Soni, Evaluating Degree of Dependency from Domain Knowledge using Fuzzy Inference System, Springer in CCIS Series, 204, pp. 101–111, 2011 CCSEIT-2011.

[29] Denzin, N. 2007. Grounded Theory and the Politics of Interpretation. Sage..

[30] A. Mauricio. Suzana P. M. M. Barrosb, Software requirements selection and prioritization using SBSE approaches: A systematic review and mapping of the literature,Journal of Systems and Software, 2014

[31] P. Achimugu ,A. Selamat, R. Ibrahim, M. N. Mahrin , A systematic literature review of software requirements prioritization research, Journal Information and Software Technology, 2014

[32] D. C. Lima, F. Freitas, G. Campos, J. Souza , A fuzzy approach to requirements prioritization. Proceedings of the 3rd SSBSE.,2011

[33] Ferreira, S., Measuring the effects of requirements volatility on software development projects, Ph.D Thesis. Department of Computer Science & Engineering, Arizona State University, Tempe, AZ., 2002.

[34] Philip A, Ali S., Roliana I., Mohd Naz'ri M., A systematic literature review of software requirements prioritization research, Information and Software Technology 56 (2014) 568–585

[35] H. Ahuja, Sujata and G. N. Purohit, "Understanding requirement prioritization techniques," 2016 International Conference on Computing, Communication and Automation (ICCCA), Noida, 2016, pp. 257-262.

[36] M. A. Abou-Elseoud, E. S. Nasr and H. A. Hefny, "Enhancing requirements prioritization based on a hybrid technique," 2016 11th International Conference on Computer Engineering & Systems (ICCES), Cairo, 2016, pp. 248-253.

[37] B. A. Mustafa and A. Zainuddin, "An experimental design to compare software requirements prioritization techniques," 2014 International Conference on Computational Science and Technology (ICCST), Kota Kinabalu, 2014, pp. 1-5

[38] M. Kobayashi, M. Maekawa, "Need-based requirements change management", Proc. 8th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems, pp. 171-178, 2001

[39] B. G. Glaser, Emergence vs forcing: Basics of grounded theory analysis: Sociology Press, 1992.

[40] Taghi J. Gandomani, Mina Z Nafchi: An empirically-developed framework for Agile transition and adoption: A Grounded Theory approach; Journal of Systems and Software; Volume 107, September 2015, Pages 204–219

[41] K. Charmaz, Discovering illness: using grounded theory", Social Science and Medicine, vol. 30, no. 11, pp. 1161-1172, 1990

[42] Parry, K.W.,1998.Grounded theory and social process : a new direction for leadership research. Leadership Quart. 9,85–105.

[43] B.G. Glaser, A.L. Strauss, The Discovery of Grounded Theory: Strategies for Qualitative Research, Aldine de Gruyter, pp. New York, 1967

[44] J. Corbin, A. Strauss, Basics of Qualitative Research Techniques and Procedures for Developing Grounded Theory, Sage, 2015.

[45] Mikael Svahnberg , Tony Gorschek , Robert Feldt , Richard Torkar , Saad Bin Saleem , Muhammad Usman Shafique, A systematic review on strategic release planning models, Information and Software Technology, v.52 n.3, p.237-248, March, 2010

[46] S. Valsala and A R Nair. Software Release Planning a Model Incorporating Environmental Parameters, Journal of Theoretical and Applied Information Technology, 68(1):4–52, 2014.

[47] D. Greer and G. Ruhe.” Software release planning: an evolutionary and iterative approach”. Information and Software Technology, 46(4):243–253, 2004

[48] Saad Bin Saleem and Muhammad Usman Shafique , “A Study on Strategic Release Planning Models of Academia and Industry Through Systematic Review and Industrial Interviews”, School of Engineering Blekinge Institute of Technology ,Sweden

[49] Laurent, P., Towards Automated Requirements Triage, 2007, IEEE, ISBN: 978-0-7695-2935-6, pp.131-140

[50] M. Yousuf, M. U. Bokhari and M. Zeyauddin, "An analysis of software requirements prioritization techniques: A detailed survey," 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, 2016, pp. 3966-3970.

[51] N.A. Zakaria, S. Ibrahim, M.N. Mahrin, using grounded theory approach to identify value-based factors in software development, 6th International Conference on Information and Communication Technology, ICT4M 2016 7814903, pp. 205-210

[52] J. R. F. D. Santos, A. B. Albuquerque and P. R. Pinheiro, "Requirements Prioritization in Market-Driven Software: A Survey Based on Large Numbers of Stakeholders and Requirements," 2016 10th International Conference on the Quality of Information and Communications Technology (QUATIC), Lisbon, 2016, pp. 67-72.

[53] S. Choochotkaew, H. Yamaguchi, T. Higashino and M. Shibuya, "Requirement-based prioritization system in multi-user IoT," 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), Milan, 2015, pp. 122-127.

[54] N. Garg, P. Agarwal and S. Khan, "Recent advancements in requirement elicitation and prioritization techniques," 2015 International Conference on Advances in Computer Engineering and Applications, Ghaziabad, 2015, pp. 237-240.

[55] https://en.wikipedia.org/wiki/PageRank

VITAE

Name	:Khalid Saad ALWAHBI
Nationality	:Saudi
Date of Birth	:12/15/1981
Email	:khalid.wahabi81@gmail.com
Address	:Saudi Arabia - Dammam
Academic Background	:BS in Computer Science